Automata Techniques for Query Inference Machines

William Gasarch[*]
University of Maryland

Geoffrey R. Hird[†]
Arcot Systems

Abstract

In prior papers the following question was considered: which classes of computable sets can be learned if queries about those sets can be asked by the learner? The answer depended on the query language chosen. In this paper we develop a framework (reductions) for studying this question. Essentially, once we have a result for queries to $[S, <]^2$, we can obtain the same result for many different languages. We obtain easier proofs of old results and several new results. An earlier result we have an easier proof of: the set of computable sets cannot be learned with queries to the language $[+, <]$ (in notation: $COMP \notin QEX[+, <]$). A new result: the set of computable sets cannot be learned with queries to the language $[+, <, \mathrm{POW}_a]$ where POW_a is the predicate that tests if a number is a power of a.

[*]Dept. of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Supported in part by NSF grants CCR-8803641 and CCR-9020079 (gasarch@cs.umd.edu).

[†]Arcot Systems, 3200 Patrick Henry Drive, Santa Clara, CA 95054. Some of the work was done when this author was at Odyssey Research Associates (geoffrey@arcot.com).

Contents

1 Introduction

In [10, 11] the following question was considered: how much can an inductive inference machine learn if it is augmented with the ability to make queries (about the computable set it is trying to learn)? The queries must be in some query language L; so L is one of the parameters of learning. The other parameters of learning are (1) the number of quantifier alternations allowed in the queries, and (2) the type of inference being used (e.g., the number of mindchanges may be bounded).

1. For certain values of the parameters, what classes of computable sets could be learned? A key theme that emerges is that the less expressive a query language is, the fewer classes of computable sets can be learned using it. Of particular interest is when $COMP$ (the class of all computable sets) can be EX-learned with queries in L; which is written as $COMP \in QEX[L]$. In [10, 11] it was shown that, for some query languages L, $COMP \notin QEX[L]$, while for others $COMP \in QEX[L]$.

2. How do the resulting learning paradigms compare to each other? In [12] it was shown that, for several query languages L, there were classes of computable sets that could be learned with no queries (just receiving data) and $n+1$ mindchanges, that could not be learned with queries to L and n mindchanges; this is written $EX_{n+1} - QEX_n[L] \neq \emptyset$. As a corollary we obtain $QEX_n[L] \subset QEX_{n+1}[L]$.

In [11] ω-regular sets were used to obtain $COMP \notin QEX[S,<]$ (queries can use symbols $<$ and S where S stands for the Successor function). In this case the machinery needed was already in the literature. In [10] the (complicated) machinery of k-good sets was developed to help obtain $COMP \notin QEX[+,<]$. In [12] k-good sets were used to show that $EX_{n+1} - QEX_n[+,<] \neq \emptyset$. In this paper we obtain $COMP \notin QEX[L]$ and $EX_{n+1} - QEX_n[L] \neq \emptyset$ for several query languages L, including $L = [+,<]$. Our proofs are considerably simpler than those in [10] and [12] and do not use k-good sets. (A recent result by Frank Stephan [19] seems to require the use of k-good sets.)

The main point of this paper is that we can now obtain results by **reducing** queries about $QEX[L]$ to queries about $QEX[S,<]^2$ (the "2" means that we can use second order quantifiers).

We now give a precise summary of our results. The non-query learning classes (those that do not begin with a Q) have been defined in [3, 18], and the query learning classes have been defined in [11]; we review these definitions in Section 2.2. The query languages mentioned are standard and are reviewed in Section 2.1.

1. Theorems about query languages in general.

   (a) Niceness. We define what it means for a query language to be nice (Definition 2.12). We then show that if L is nice then $(\forall a, n)[PEX_{n+1} - QEX_n^a[L] \neq \emptyset]$ (Theorem 3.1).

   (b) Reductions. We define two notions of reduction $L \leq L^\$$ and $L \leq_w L^\$$ (Definitions 4.2 and 4.5). We also define two refinements of them $L \leq_{\mathsf{N}} L^\$$ and $L \leq_{wb} L^\$$ (Definitions 4.3 and 4.6). Let $a, d, n \in \mathsf{N}$ with $d \geq 1$. We show the following.

      i. $L \leq_{\mathsf{N}} [S, <]^2 \Rightarrow COMP \notin QBC^a[L]$.
      ii. $L \leq_{\mathsf{N}} [S, <]^2 \Rightarrow COMP \notin [1, d]QBC^a[L]$.
      iii. $L \leq_{\mathsf{N}} [S, <]^2 \Rightarrow PEX_{n+1} - QEX_n^a[L] \neq \emptyset$.
      iv. $L \leq_{\mathsf{N}} [S, <]^2 \Rightarrow QEX_n^a[L] \subset QEX_{n+1}^a[L]$ (easy corollary).
      v. $L \leq_{wb} [S, <]^2 \Rightarrow PEX_{n+1} - QEX_n^a[L] \neq \emptyset$.
      vi. $L \leq_{wb} [S, <]^2 \Rightarrow QEX_n^a[L] \subset QEX_{n+1}^a[L]$ (easy corollary).
      vii. $L \leq_{wb} [S, <]^2 \Rightarrow L$ is QCD. (QCD is a decidability criteria on languages. It will be defined in Section 2.3.)

2. Theorems about particular query languages. Let $b \geq 2$.

   (a) $[+, <] \leq_{\mathsf{N}} [S, <]^2$.

   (b) $[+, <, \mathrm{POW}_b] \leq_{\mathsf{N}} [S, <]^2$.

   (c) For a large class of predicates P, $[S, <, P]^2 \leq_{wb} [S, <]^2$. The class of predicates P includes FAC, POW_b, and POLY_b.

   (d) Using $1b$ and $2a, 2b, 2c$ we obtain many results about query inference classes.

For the reader who just wants to see the easier proof of $COMP \notin QEX[+, <]$ do the following: Read the proof that $COMP \notin QEX[S, <]$ from [11] and adjust it to work for $COMP \notin QEX[S,<]^2$ (this is easy). Then read Theorem 4.4.3 and Lemma 6.1 of this paper.

Some of the ideas in this paper are similar to those of Semenov [17].

2 Definitions

2.1 Query Languages

For this paper we will only consider learning classes of computable sets (as opposed to functions). Hence in the definitions below the queries are about sets. This restriction does not affect our results since, for example, if the set of computable sets cannot be inferred by some machine, then certainly the set of all computable functions cannot either.

Definition 2.1 A *query language* consists of the usual logical symbols (and equality), symbols for first and second order variables, symbols for every element of N, and symbols for some functions and relations on N. A query language is denoted by the symbols for these functions and relations. We will use a superscript 2 to indicate that we allow quantification over second order variables. For example we refer to 'the query language $[+,<]$' or 'the query language $[S,<]^2$.' A well-formed formula over L is defined in the usual way.

Convention 2.2 Small letters are used for first order variables which range over N. Capital letters are used for second order variables which range over subsets of N.

Definition 2.3 Let L be a query language. A *query over* L is a formula $\phi(X)$ such that the following hold.

1. $\phi(X)$ uses symbols from L.
2. X is a free set variable and is the only free variable in ϕ.
3. If L does not allow quantification over second order variables then X (which is not quantified over) is the only second order variable in $\phi(X)$.

We think of a query $\phi(X)$ as asking a question about an as yet unspecified set $X \subseteq \mathsf{N}$. If $A \subseteq \mathsf{N}$ then $\phi(A)$ will be either true or false.

Notation 2.4 Let $b \in \mathsf{N}$, $b \geq 2$. We will be using the following symbols in some of our query languages

1. S stands for the successor function $S(x) = x + 1$.
2. POW_b is the unary predicate that determines if a number is in the set $\{b^n : n \in \mathsf{N}\}$.
3. POLY_b is the unary predicate that determines if a number is in the set $\{n^b : n \in \mathsf{N}\}$.
4. FAC is the unary predicate that determines if a number is in the set $\{n! : n \in \mathsf{N}\}$.

2.2 Inductive Inference

We briefly review concepts from both passive and query inference. For a fuller treatment see Case and Smith [3], and Smith [18], for passive inference; and see Gasarch and Smith [11] for query inference. We will only be concerned with learning classes of computable sets and our definitions will reflect this.

Notation 2.5 We assume that Turing machines are coded into natural numbers in some fixed computable fashion. We assume that all of our Turing machines compute partial computable sets (on all inputs either output 0 or 1 or diverge). Let φ_e be the partial computable set computed by the Turing machine that is coded by e. We refer to e as a *program.*

An inductive inference machine (IIM) is a total Turing machine M. We interpret M as trying to learn a computable set A as follows. M is presented with the values $A(0)$, $A(1)$, $A(2), \ldots$ and will, over time, output conjectures e indicating that M thinks A is decided by φ_e. (φ_e need not be total. In this case M's guess is wrong.) M has no other way of obtaining additional information about A. A query inference machine (QIM) is a total Turing machine that can make queries about the computable set A in a particular query language (and magically get the answers). Note that a QIM gets all of its information from making queries and does not see any data; however it can request whatever data it wants (i.e., the QIM can ask '$17 \in A$?').

Let M be either an IIM or a QIM. Both query and passive inductive inference deal with inference in the limit; that is, we think of M as executing forever. From time to time, M may make a conjecture about the set A, in the form of a program e such that M thinks (at least for now) that A is decided by φ_e. The guess can be null, denoted $\perp$. If M guesses the same program from some point on, and that program is correct, then we say that *M EX-inferred* the set A. A *concept class* is a subset of $COMP$; we say that *M EX-infers the concept class $\mathcal{S}$* if it EX-infers every $A \in \mathcal{S}$. EX is the set of concept classes which are inferred by some IIM. The term EX stands for 'explains.' The idea is that we are looking at data and we wish to explain it by producing an index for a machine that behaves just like the data. (This motivation is from [3].)

$QEX[L]$ is the set of concept classes that are inferred by some QIM that makes queries in the query language L. For $i \in \mathsf{N}$, $Q_iEX[L]$ is the set of concept classes of computable sets which can be inferred in the limit by a QIM that makes queries in the query language L, where the queries are restricted to i alternations of quantifiers.

We define several variations on inference. Let M be an IIM, A be a computable set, $c, d, n \in \mathsf{N}$ with $c, d \geq 1$, and $a \in \mathsf{N} \cup \{*\}$.

1. *Mindchanges* [3]. If M infers A and makes $\leq n+1$ different conjectures then *M EX_n-infers A*. $\mathcal{S} \in EX_n$ if there is an IIM M such that, for every $A \in \mathcal{S}$, M EX_n-infers A. $QEX_n[L]$ and $Q_iEX_n[L]$ can be defined similarly. (The name 'mindchange' comes from the fact that if $n+1$ different programs are output then the conjecture changes at most n times. The change from $\perp$ to a non-null guess is not counted.)

2. *Anomalies* [3]. Let $a \in \mathsf{N}$. If φ_e and A differ on at most a points then *φ_e is an a-variant of A*. If M is trying to infer A and, from some point on, always outputs e where φ_e is an a-variant of A, then *M EX^a-infers A*. If A and φ_e differ on at most a finite number points then φ_e is *is a finite variant of A*. If M is trying to infer A and, from some point on, always outputs e where φ_e is a finite variant of A then *M EX^*-infers A*. In either case ($a \in \mathsf{N}$ or $a = *$) $\mathcal{S} \in EX^a$ if there exists an IIM M such that, for every $A \in \mathcal{S}$, M EX^a-infers A. $QEX^a[L]$ and $Q_iEX^a[L]$ can be defined similarly. $QEX^a[L]$ was studied in [8].

3. *Teams* [14, 18], Let $M_1, \ldots, M_d$ be a set of d IIMs. If at least c of $\{M_1, \ldots, M_d\}$ correctly EX-infer A then A *is* $[c,d]$*-inferred by* $\{M_1, \ldots, M_d\}$. $\mathcal{S} \in [c,d]EX$ if there exists $M_1, \ldots, M_d$ such that, for every $A \in \mathcal{S}$, A is $[c,d]$*-inferred by* $\{M_1, \ldots, M_d\}$. The set of machines $\{M_1, \ldots, M_d\}$ is referred to as a *team*. $[c,d]QEX[L]$ and $[c,d]Q_iEX[L]$ can be defined similarly. $[c,d]QEX[L]$ was studied in [8]. Note that different sets in $\mathcal{S}$ may be inferred by different size-c subsets of $M_1, \ldots, M_d$.

4. *Popperian Inference* [3]. $\mathcal{S} \in PEX$ if $\mathcal{S} \in EX$ via an IIM that only conjectures total programs. (By contrast, the guesses made by a machine for EX inference may be non-total.) This concept was studied and named in [3]. $[c,d]QPEX[L]$ and $Q_iPEX[L]$ can be defined similarly. $QPEX[L]$ was studied in [12].

5. *Behaviorally Correct* [3]. M *BC-infers* A if, when M is fed A, eventually all of M's guesses about A are indices e where $\varphi_e = A$. This differs from EX in that if M EX-infers A then past some point all the guesses are *the same* and decide A, where as here they need not be the same, they need only decide A. BC is the set of concept classes of computable sets which can be behaviorally inferred by some IIM M. If M is trying to infer A and, from some point on, always outputs an index for a program that is an a-variant of A, then M BC^a*-infers* A. The classes BC^a, $QBC[L]$, and $Q_iBC[L]$ can all be defined easily.

6. *Combinations.* The parameters above can be combined to yield classes like $[c,d]EX_n^a$. Comparisons between these classes have been studied extensively [7]. We will deal with the combinations PEX_n, PEX_n^a, QEX_n^a, and $[1,d]QBC^a[L]$. We may use phrases like "M $Q_iEX_n^a$-infers A" which should be understood. (We do not consider BC^* since $COMP \in BC^*$. Case and Smith present Harrington's proof of this in [3].)

Definition 2.6 *COMP is the class of all computable subsets of* N*. Let* $E \in COMP$*.* $COMP_E$ *is the set of all computable subsets of* E*.* $COMP$ *is the largest possible concept class of sets.*

It is known that $COMP \in QEX[+,\times]$ (in fact $COMP \in Q_1EX[+,\times]$ and $COMP \in Q_2EX_0[+,\times]$ [11, Theorem 9,10]). In [11, Theorem 23] it was shown that $COMP \notin QEX[S,<]$. In [10] it was shown that $COMP \notin QEX[+,<]$. In [12] it was shown that, for all n, $PEX_{n+1} - QEX_n[+,<] \neq \emptyset$. These last two results had rather complicated proofs.

In this paper we will often show a result about team learning and then obtain from that a result about learning with anomalies via the following lemma. The proof is similar to Theorem 6.1 of [18].

Lemma 2.7 *Let $a, d \in \mathsf{N}$. Let $\mathcal{I}$ be any of $\{EX, BC, EX_m\}$. Let $\mathcal{I}^a$ be the notion $\mathcal{I}$ allowing at most a anomalies. Let $Q\mathcal{I}$ and $Q\mathcal{I}^a$ be the corresponding query notions. Then $[1,d]Q\mathcal{I}^a[L] \subseteq [1, d(a+1)]Q\mathcal{I}[L]$.*

2.3 Variants of Second Order Decidability

We define the notions of *Cardinality Decidable* and *Quasi Cardinality Decidable.* These notions played an important role in the proofs that $COMP \notin QEX[S,<]$ and $COMP \notin QEX[+,<]$, and are also of independent interest. As an easy consequence of our reductions we will obtain that several query languages are quasi cardinality decidable.

Definition 2.8 We identify a query $\phi(X)$ with the set $\{A : \phi(A)\}$. Hence a query ϕ is *uncountable* if $\{A : \phi(A)\}$ is uncountable. Let E be an infinite set. A query $\phi(X)$ is *E-uncountable* if $\{A \subseteq E : \phi(A)\}$ is uncountable. We abbreviate this E-unc. Note that the information that $\phi(X)$ is E-unc does not tell us anything about $\neg\phi(X)$ being E-unc or N-unc.

Definition 2.9 Let L be a query language. Let E be an infinite subset of N.

1. L is *E-cardinality decidable* (abbreviated *E-CD*) if the following set is decidable: $\{\psi : \psi(X) \text{ is } E\text{-unc}\}$. L is CD if it is N-CD.

2. L is *quasi cardinality decidable* (abbreviated QCD) if there is a partial computable function which behaves as follows. If ψ is a query then the function returns $\phi \in \{\psi, \neg\psi\}$ such that ϕ is uncountable. Note that even though we know that ϕ is uncountable we do not know if $\neg\phi$ is uncountable.

Using properties of ω-regular sets it is easy to show that $[S,<]$ is CD (see [11, Lemma 21]). It is known [6] that $[+,<]$ is not second order decidable and not CD. However, in [10] it was shown that $[+,<]$ is QCD. Hence even if a language is not second order decidable it may still be QCD.

The proof that $[+,<]$ is QCD was somewhat difficult in that it used the machinery of k-good sets. In this paper we will prove that $[S,<]^2$ is CD (easily) and from that show that $[+,<]$ and several other query languages are QCD. These proofs will be easier than those in [10] in that they avoid the use of k-good sets.

2.4 Nice Query Languages

In this section we define what it means for a query language to be nice. The definition will be unnatural. However it will be just what we need because simultaneously (1) we will be able to prove theorems in inductive inference for nice query languages, and (in Section 3) (2) we will be able to prove that several query languages are nice.

Definition 2.10 Let $a \in \mathsf{N}$. If A and B are sets then $A =^{=a} B$ means that A and B (i.e., their characteristic sequences) differ on *exactly* a places.

We give the intuition behind the next definition. Imagine that there is a set W such that, for any query ψ, if $\psi(W)$ holds then there exists a set W_1 that is not too much different from W (e.g., $W =^{=c} W_1$) such that $\psi(W_1)$ also holds. Imagine that W_1 has the same property— for any query ψ, if $\psi(W_1)$ holds then there exists a set W_2 that is not too much different from W_1 (e.g., $W =^{=c} W_1 =^{=c} W_2$ in such a way that $W =^{2c} W_2$) such that $\psi(W_2)$ holds. Imagine that this goes on for $W_3, W_4, \ldots$. Such a set W would be useful in constructions since we could feed a QIM answers that are true of W and then later switch to W_1, and later to W_2, etc. Intuitively, a language is nice if such a W exists. We now formalize this notion.

Definition 2.11 Let L be a query language. Let E be an infinite subset of N, $W \subseteq E$, and $c \in \mathsf{N}$. L is (E,W,c)-*nice* if the following holds. For all queries ψ and $e \in \mathsf{N}$

$$(\exists W' \subseteq E)[\psi(W') \wedge (W' =^{=ce} W)] \Rightarrow (\exists W'' \subseteq E)[\psi(W'') \wedge (W'' =^{=ce+c} W)].$$

Definition 2.12 L is *E-nice* if there exists (W, c) such that W is computable and L is (E, W, c)-nice. L is *nice* if there exists (E, W, c) such that W is computable and L is (E, W, c)-nice.

In [12] it was shown (implicitly) that $[+, <]$ is $(\mathsf{N}, ACK, 1)$-nice where ACK is the range of a fast growing function (e.g., a 1-variable version of Ackerman's function). This proof was somewhat difficult in that it used the machinery of k-good sets from [10]. It was also the first proof that $[S, <]$ is nice. In this paper we will show $[S, <]^2$ is nice (see Lemma 5.22) and then **derive from this using reductions** that several other query languages, including $[+, <]$, are nice. Our proofs are easier than those in [12] in that we use ω-regular sets instead of k-good sets.

2.5 Convention and Definition

The following convention and definition are used throughout this paper.

Convention 2.13 Let L be a query language. Let $\phi_1, \phi_2, \phi_3, \ldots$ be all possible queries in that query language in some fixed computable order. We will assume that if M is a query inference machine that is using query language L then the ith query M makes is ϕ_i. This entails no loss of generality since all queries the machine wanted to make are eventually made.

Definition 2.14 Let L be a query language, $t \in \mathsf{N}$, and $\vec{c} \in \{0, 1\}^t$. We think of $\vec{c}$ as answers to the first t queries. For $1 \le i \le t$ let

$$\psi_i = \begin{cases} \phi_i & \text{if } \vec{c}[i] = 1; \\ \neg\phi_i & \text{if } \vec{c}[i] = 0. \end{cases}$$

Let $\psi = \bigwedge_{i=1}^{t} \psi_i$. ψ is said to *capture the answers $\vec{c}$ to the first t queries.* If $L = [S, <]^2$ then an automaton $\mathcal{A}$ *captures $\vec{c}$* if $(\forall X)[\mathcal{A}(X) \Rightarrow \bigwedge_{i \le t} \psi_i(X)]$.

3 L Nice $\Rightarrow (\forall a, n)[PEX_{n+1} - QEX_n^a[L] \neq \emptyset]$

Theorem 3.1 *If L is nice then $(\forall n)[PEX_{n+1} - QEX_n[L] \neq \emptyset]$. Moreover, if L is (E, W, c)-nice with a computable W then*

$$(\forall n)(\exists \mathcal{S} \subseteq COMP_E)[\mathcal{S} \in PEX_{n+1} - QEX_n[L]]$$

where $COMP_E$ is the class of all computable subsets of E (see Definition 2.6).

Proof:

Let L be (E, W, c)-nice with W computable. Let

$$\mathcal{S} = \bigcup_{i=0}^{n+1} \{W' \subseteq E : W' =^{=ci} W\}.$$

(See Definition 2.10 for the definition of $=^{=ci}$.) Note that since W is computable $\mathcal{S} \subseteq COMP_E$. We show that $\mathcal{S} \in PEX_{n+1} - QEX_n[L]$. $\mathcal{S}$ is in PEX_{n+1} by the following process. (Let A denote the set being observed. Note that A and W differ on a multiple of c points.)

1. Initially output e_0, an index for W. Set $i = 0$. φ_{e_i} and W differ in a multiple of c places. Since $A \in \mathcal{S}$, A and W differ in a multiple of c places. Hence A and φ_{e_i} differ in a multiple of c places. This will be true inductively.

2. As long as A agrees with φ_{e_i} we keep outputting e_i. This can be checked since φ_{e_i} will always be total.

3. If an x is observed such that $A(x) \neq \varphi_{e_i}(x)$ then we do the following.

   (a) Keep observing A until c points $a = a_1, \ldots, a_c$ are found such that $(\forall j)[f(a_j) \neq \varphi_{e_i}(a_j)]$. (This must happen since A and φ_{e_i} differ on a multiple of c places, and we have just observed one more place where A and φ_{e_i} differ.)

   (b) Let g be defined by

   $$B(x) = \begin{cases} \varphi_{e_i}(x) & \text{if } x \notin \{a_1, \ldots, a_c\}; \\ A(x) & \text{if } x \in \{a_1, \ldots, a_c\}. \end{cases}$$

   Set e_{i+1} to be a program for B. Let $i = i + 1$.

   (c) Go to step 2.

Assume $A \in \mathcal{S}$ and i is such that $A =^{=ci} W$ where $i \le n+1$. Clearly A will be properly PEX-inferred with $i \le n+1$ mindchanges. Hence $\mathcal{S} \in PEX_{n+1}$.

We show that $\mathcal{S} \notin QEX_n[L]$. Assume, by way of contradiction, that $\mathcal{S} \in QEX_n[L]$ via QIM M. We construct a set $A \in \mathcal{S}$ that M does not infer. Our construction is non-effective; however the set produced in the end is in $\mathcal{S}$.

BEGIN CONSTRUCTION

1. Let $A_0 = W$, $i = 0$. Note that $A_i =^{=ic} W$, $A_0 \subseteq E$, and $A_i \in \mathcal{S}$. These statements will be true throughout.

2. If $i = 0$ then start the simulation of M on A_i. If $i > 0$ then continue the simulation. (We will later see that this is possible.) Answer queries as though A_i is the set being inferred. Stop when a conjecture e_i is made such that φ_{e_i} decides A_i. (This must happen since $A_i \in \mathcal{S}$. This step is highly nonconstructive in both answering queries and in knowing when to stop.)

3. Let ψ be the query that captures all the query answers thus far (see Definition 2.14). Because of how part 2 executed we have $\psi(A_i)$. Inductively we have $A_i =^{=ci} W$ and $A_i \subseteq E$. Since L is (E, W, c)-nice there exists W'' such that $W'' =^{=ci+c} W$ and $\psi(W'')$. Let $A_{i+1} = W''$. Note that φ_{e_i} does not decide A_{i+1}.

4. There are two cases.

   (a) If $i < n$ then let $i = i + 1$ and go to step 2. Note that we have $A_i =^{=ic} W$, $A_0 \subseteq E$, and $A_i \in \mathcal{S}$. Also note that we have $\psi(A_i)$; hence the simulation in step 2 can continue.

   (b) If $i = n$ then n mindchanges have occurred so no more queries can be made. Let A be A_{n+1} and terminate the construction.

END OF CONSTRUCTION

The following can be proven inductively. If $0 \le i \le n$ then, at the end of step 3, $A_{i+1} =^{=ci+c} W$ and φ_{e_i} does not decide A_{i+1}. Hence $A = A_{n+1} \in \mathcal{S}$. Note that if M tries to infer A then its sequence of guesses is $e_0, e_1, \ldots, e_n$. Since φ_{e_n} does not decide A, M does not infer A. ▮

Theorem 3.2 *If L is nice then $(\forall a,n)[PEX_{n+1} - QEX_n^a[L] \neq \emptyset]$, and hence (easily) $QEX_n^a[L] \subset QEX_{n+1}^a[L]$. Moreover, if L is (E,W,c)-nice with a computable W then $(\forall a,n)(\exists \mathcal{S} \subseteq COMP_E)[\mathcal{S} \in PEX_{n+1} - QEX_n^a[L]]$.*

Proof: Let L be (E,W,c)-nice with W computable. By iterating the definition of niceness we can assume that $c \geq 2a+1$. Let

$$\mathcal{S} = \bigcup_{i=0}^{n+1} \{W' \subseteq E : W' =^{=ci} W\}.$$

Note that since W is computable $\mathcal{S} \subseteq COMP_E$. $\mathcal{S} \in PEX_{n+1} - QEX_n[L]$ as in the proof of Theorem 3.1.

The construction to show $\mathcal{S} \notin QEX_n^a[L]$ is identical to the one above except that we wait for a conjecture e_i such that φ_{e_i} is an a-variant of A_i. Note that A_i and A_{i+1} differ in exactly $c \geq 2a+1$ places. Since φ_{e_i} decided an a-variant of A_i, φ_{e_i} must differ from A_{i+1} in at least $c-a \geq a+1$ places. Hence a new conjecture is required. ▮

4 Reductions

In this section we define reductions $\leq$ and $\leq_w$ between query languages L and $L^{\$}$. We will also define two refinements of them, $\leq_{\mathsf{N}}$ and $\leq_{wb}$ respectively, which will be the ones we actually use.

4.1 $L \leq L^{\$}$ and $L \leq_{\mathsf{N}} L^{\$}$

Notation 4.1 Let E, $E^{\$}$ be infinite subsets of N, and f be a bijection between E and $E^{\$}$. We extend the definition of f to $\mathcal{P}(E)$ and to E^* by $f(A) = \{f(n) : n \in A\}$ and $f(n_1,\ldots,n_a) = (f(n_1),\ldots,f(n_a))$. Note that f (and its extensions) is a bijection between E and $E^{\$}$ ($\mathcal{P}(E)$ and $\mathcal{P}(E^{\$})$, E^* and ${E^{\$}}^*$). We will use f for all three of these functions. The meaning will be clear from context.

Definition 4.2 Let L, $L^{\$}$ be query languages and E, $E^{\$}$ be infinite computable subsets of N. Let f be a computable bijection from E to $E^{\$}$. Formally everything in this definition is parameterized by E, $E^{\$}$, and f; however we will not make this explicit. L *is reducible to* $L^\$$, written $L \le L^\$$, if there exists a computable function with the following properties.

1. The input is a query $\psi(X)$ over L and the output is a query $\psi^\$(X)$ over $L^\$$. (This is called *the domain condition.*)
2. $(\forall A \subseteq E)[\psi(A) \text{ iff } \psi^\$(f(A))]$. (This is called *the equivalence condition.*)

We will be most interested in reductions when $E^\$ = \mathsf{N}$.

Definition 4.3 Let L, $L^\$$ be query languages. $L \le_{\mathsf{N}} L^\$$ if there exists (E, f) such that $L \le L^\$$ with parameters (E, N, f). (Note that since f is a computable bijection and N is computable, E is forced to be computable.)

The following theorem will be used in Section 5.1.

Theorem 4.4 *If $L \le L^\$$ with parameters $(E, E^\$, f)$ then the following hold.*

1. *Let $\mathcal{I}, \mathcal{J}$ be any of the passive notions of inference discussed in Section 2.2. Let $Q\mathcal{I}$ be the corresponding query notion. For all $\mathcal{A} \subseteq COMP_E$ the following hold.*
   (a) *$\mathcal{A} \in \mathcal{I}$ iff $f(\mathcal{A}) \in \mathcal{I}$.*
   (b) *$\mathcal{A} \in Q\mathcal{I}[L] \Rightarrow f(\mathcal{A}) \in Q\mathcal{I}[L^\$]$.*
   (c) *$COMP_E \in Q\mathcal{I}[L] \Rightarrow COMP_{E^\$} \in Q\mathcal{I}[L^\$]$.*
   (d) *If $E^\$ = \mathsf{N}$ (so $L \le_{\mathsf{N}} L^\$$ by definition) and $COMP \notin Q\mathcal{I}[L^\$]$ then $COMP \notin Q\mathcal{I}[L]$.*
2. *If $\mathcal{I} - Q\mathcal{I}[L^\$] \ne \emptyset$ then $\mathcal{I} - Q\mathcal{I}[L] \ne \emptyset$.*
3. *If $E^\$ = \mathsf{N}$ and $COMP \notin Q\mathcal{I}[L^\$]$ then $COMP \notin Q\mathcal{I}[L]$.*
4. *If $U, V \subseteq E$ then $[U =^{=a} V$ iff $f(U) =^{=a} f(V)]$. (See Definition 2.10 for the definition of $=^{=a}$.)*
5. *If $L^\$$ is nice then L is nice.*
6. *If $(\forall a, n)[PEX_{n+1} - QEX_n^a[L^\$] \ne \emptyset]$ then $(\forall a, n)[PEX_{n+1} - QEX_n^a[L] \ne \emptyset]$.*

7. *If ψ is a query over L then [ψ is E-unc iff $\psi^{\$}$ is $E^{\$}$-unc].*

8. *If $L^{\$}$ is $E^{\$}$-$CD$ then L is QCD.*

Proof:

Throughout this proof ψ will denote a query over L and $\psi^{\$}$ will denote the corresponding (using $L \le L^{\$}$) query over $L^{\$}$. Note that from ψ we can effectively obtain $\psi^{\$}$.

1a) This is obvious since f and f^{-1} are both computable.

1b) We prove this for $QEX[L]$. The other proofs are similar. Assume $\mathcal{A} \subseteq COMP_E$ and $\mathcal{A} \in QEX[L]$ via M. The following inference procedure shows $f(\mathcal{A}) \in QEX[L^{\$}]$.

ALGORITHM

```
1. (We are inferring $f(A) \subseteq E^{\$}$.) Let $\vec{b} = \lambda$.
2. Let $\psi$ be the query $M$ asks upon seeing the vector $\vec{b}$ of answers.
3. Ask $\psi^{\$}$ and let $b \in \{0,1\}$ be the answer. Note that $\psi(A)$ iff $\psi^{\$}(f(A))$.
   Let $\vec{b}$ be $\vec{b} \cdot b$.
4. Let $e$ be the conjecture made by $M$ when seeing vector $\vec{b}$. Output an
   index for the program $\varphi_e \circ f^{-1}$.
```

END OF ALGORITHM

Since in the limit e stabilizes and φ_e decides A, we have that the output of this process stabilizes and the output decides $f(A)$.

1c) This follows from 1b since $f(COMP_E) = COMP_{E^{\$}}$.

1d) $COMP \in Q\mathcal{I}[L] \Rightarrow COMP_E \in Q\mathcal{I}[L] \Rightarrow COMP = COMP_{E^{\$}} \in Q\mathcal{I}[L^{\$}]$.

2) This follows from 1a and 1b

3) This follows from 1c and 1d

4) If $U =^{=a} V$ then $f(U) =^{=a} f(V)$ since f is a bijection.

5) Assume $L^{\$}$ is $(E^{\$}, W^{\$}, c)$-nice. Let $W = f^{-1}(W^{\$}) \subseteq E$. Note that since $W^{\$}, E$, and $E^{\$}$ are computable, W is computable. We show that L is (E, W, c)-nice.

Let $W' \subseteq E$, $W' =^{=ce} W$, and $\psi(W')$. By part 2 and the definition of reduction

$$(f(W') =^{=ce} f(W) = W^{\$}) \wedge \psi^{\$}(f(W')).$$

Since $L^{\$}$ is $(E^{\$}, W^{\$}, c)$-nice

$$(\exists W^{\$\prime\prime})[\psi^{\$}(W^{\$\prime\prime}) \wedge (W^{\$\prime\prime} =^{=ce+c} W^{\$})].$$

By part 2 and the definition of reduction

$$\psi(f^{-1}(W^{\$\prime\prime})) \wedge (f^{-1}(W^{\$\prime\prime}) =^{=ce+c} f^{-1}(W^{\$}) = W) \wedge (f^{-1}(W^{\$\prime\prime}) \subseteq E).$$

Hence $W'' = f^{-1}(W^{\$\prime\prime})$ will suffice.

6) This follows from 5 and Theorem 3.2.

7) Let ψ be a query over L. f restricted to $\{A : A \subseteq E \wedge \psi(A)\}$ is a bijection between $\{A : A \subseteq E \wedge \psi(A)\}$ and $\{A : A \subseteq E^{\$} \wedge \psi^{\$}(A)\}$. The conclusion follows.

8) Assume $L^{\$}$ is $E^{\$}$-*CD*. We show L is *QCD*. Given a query ψ over L we know that either ψ or $\neg\psi$ is E-unc. Hence at least one of $\{\psi^{\$}, (\neg\psi)^{\$}\}$ is $E^{\$}$-unc. Using that $L^{\$}$ is $E^{\$}$-*CD* we can determine which of these is $E^{\$}$-unc (it could be both). If it is $\psi^{\$}$ then output ψ, else output $\neg\psi$. (Note that we only get partial information— if a formula $\phi^{\$}$ is *not* $E^{\$}$-uncountable, we are not entitled to conclude that ϕ is N-countable.) ▮

4.2 $L \leq_w L^{\$}$ and $L \leq_{wb} L^{\$}$

In the definition of $\leq$ we fixed E,$E^{\$}$, and f. In the next definition we let $E^{\$}$ and f vary to an extent. The resulting definition is weaker and will not yield a theorem as strong as Theorem 4.4.

Definition 4.5 Let L, $L^{\$}$ be query languages and E be an infinite computable subset of N. Let $\{E^{\$}_d\}_{d=1}^{\infty}$ be a sequence of infinite computable subsets of N, and $\{f_d\}_{d=1}^{\infty}$ be such that f_d is a computable bijection between E and $E^{\$}_d$. We assume that from d we can obtain Turing machine indices for $E^{\$}_d$ and f_d. Formally everything in this definition is parameterized by E, $\{E^{\$}_d\}_{d=1}^{\infty}$, and $\{f_d\}_{d=1}^{\infty}$, however we will not make this explicit. *L is weakly reducible to $L^{\$}$*, written $L \leq_w L^{\$}$, if there exists a computable function with the following properties.

1. The input is a query $\psi(X)$ over L. The output is a pair: a query $\psi^{\$}(X)$ over $L^{\$}$, and a number d (indicating that we are concerned with $E_d^{\$}$).
2. $(\forall A \subseteq E)[\psi(A) \text{ iff } \psi^{\$}(f_d(A))]$.

We will be most interested in reductions when the singleton sets $\{E_d^{\$}\}$ are ω-regular.

Definition 4.6 The sequence of sets $\{E_d^{\$}\}_{d=1}^{\infty}$ is *well behaved* if the following are true.

1. For each d, the singleton set $\{E_d^{\$}\}$ (viewed as an element of $\{0,1\}^{\omega}$) is ω-regular.
2. There is an effective procedure to go from d to an Büchi-automaton for $\{E_d^{\$}\}$.

Let L and $L^{\$}$ be query languages. $L \le_{wb} L^{\$}$ if there exists parameters $(E, \{E_d^{\$}\}_{d=1}^{\infty}, \{f_d\}_{d=1}^{\infty})$ such that $L \le_w L^{\$}$ with these parameters and $\{E_d^{\$}\}_{d=1}^{\infty}$ is well behaved. (The wb stands for well behaved.)

Definition 4.7 Let $L^{\$}$ and $\{E_d^{\$}\}_{d=1}^{\infty}$ be as in Definition 4.5. $L^{\$}$ *is uniformly $E_d^{\$}$-CD* if there exists a partial computable function which, on input (ψ, d) determines if ψ is $E_d^{\$}$-unc.

Theorem 4.8 *Let $(L, L^{\$}, E, \{E_d^{\$}\}_{d=1}^{\infty}, \{f_d\}_{d=1}^{\infty})$ be as in Definition 4.5. Assume $L \le_w L^{\$}$.*

1. $(\forall d)[U, V \subseteq E \Rightarrow [U =^{=a} V$ *iff* $f_d(U) =^{=a} f_d(V)]]$. *(See Definition 2.10 for the definition of $=^{=a}$.)*
2. *If there exists $c \in \mathsf{N}$, sets $\{W_d^{\$}\}_{d=1}^{\infty}$, and W such that for every d*
   (a) *$W_d^{\$}$ is a computable subset of $E_d^{\$}$,*
   (b) *$L^{\$}$ is $(E_d^{\$}, W_d^{\$}, c)$-nice,*
   (c) *$f_d^{-1}(W_d^{\$}) = W$ (so all the $W_d^{\$}$ map back to the same set W)*

   then L is (E, W, c)-nice.

3. *If* $(\forall a, n)[PEX_{n+1} - QEX_n^a[L^\$] \neq \emptyset]$ *then* $(\forall a, n)[PEX_{n+1} - QEX_n^a[L] \neq \emptyset]$.

4. *If* ψ *is a query over* L *and* ψ *maps to* $(\psi^\$, d)$ *then* [ψ *is* E*-unc iff* $\psi^\$$ *is* $E_d^\$$*-unc*].

5. *If* $L^\$$ *is uniformly* $E_d^\$$*-CD then* L *is QCD.*

Proof:

The proofs of parts 1,4,5 are similar to the proofs of parts 2,6,7 of Theorem 4.4 and are hence omitted. We prove parts 2 and 3.

2) Since $W^\$, E$, and $E_d^\$$ are computable, W is computable. We show that L is (E, W, c)-nice.

Let $W' \subseteq E$, $W' =^{=ce} W$, and $\psi(W')$. Let ψ get mapped to $(\psi^\$, d)$. By part 1 and the definition of reduction

$$(f_d(W') =^{=ce} f_d(W) = W_d^\$) \wedge \psi^\$(f(W')).$$

Since $L^\$$ is $(E^\$, W_d^\$, c)$-nice

$$(\exists W_d^{\$''})[\psi^\$(W_d^{\$''}) \wedge (W_d^{\$''} =^{=ce+c} W_d^\$)].$$

By part 1 and the definition of reduction

$$\psi(f_d^{-1}(W_d^{\$''})) \wedge (f_d^{-1}(W_d^{\$''}) =^{=ce+c} f_d^{-1}(W_d^\$) = W) \wedge (f_d^{-1}(W_d^{\$''}) \subseteq E).$$

Note that $f^{-1}(W_d^{\$''}) \subseteq E$. Hence $W'' = f^{-1}(W_d^{\$''})$ will suffice.

3) This follows from 2 and Theorem 3.2.

▮

5 Consequences of $L \leq [S, <]^2$

In this section we show that, for all $a, d, n \in \mathsf{N}$, with $d \geq 1$, the following hold.

1. $L \leq_{\mathsf{N}} [S, <]^2 \Rightarrow COMP \notin QBC^a[L]$.

2. $L \leq_{\mathsf{N}} [S, <]^2 \Rightarrow COMP \notin [1, d]QBC^a[L]$.

3. $L \leq_{\mathsf{N}} [S,<]^2 \Rightarrow PEX_{n+1} - QEX_n^a[L] \neq \emptyset \Rightarrow QEX_n^a[L] \subset QEX_{n+1}^a[L]$.
4. $L \leq_{wb} [S,<]^2 \Rightarrow PEX_{n+1} - QEX_n^a[L] \neq \emptyset \Rightarrow QEX_n^a[L] \subset QEX_{n+1}^a[L]$.
5. $L \leq_{wb} [S,<]^2 \Rightarrow L$ is QCD.

Throughout this section $a, d, n \in \mathsf{N}$, $d \geq 1$, are fixed.

5.1 $L \leq_{\mathsf{N}} [S,<]^2 \Rightarrow COMP \notin QBC[L]$

We first show that $COMP \notin QBC[S,<]^2$ and then use Theorem 4.4.3 to easily obtain that, for all L, $L \leq_{\mathsf{N}} [S,<]^2 \Rightarrow COMP \notin QBC[L]$.

While proving $COMP \notin QBC[S,<]^2$ we will need to pick points to diagonalize on very carefully. The next subsection contains lemmas that will help us to do that.

5.1.1 ω-Regular Sets

The language $L = [S,<]^2$ is related to ω-regular sets. We review this relation. See [20] for a survey of automata on infinite objects and their relation to logic.

Notation 5.1 Let Σ be a finite alphabet.

1. If $V \subseteq \Sigma^*$ then let V^ω be the set of all infinite sequences of nonempty strings chosen from V.
2. If $\sigma \in \Sigma^*$ and $\tau \in \Sigma^* \cup \Sigma^\omega$ then $\sigma \preceq \tau$ means that σ is a prefix of τ, and $\sigma \prec \tau$ means that σ is a proper prefix of τ
3. If $A \subseteq \mathsf{N}$ and $\sigma \in \{0,1\}^*$ then $\sigma \prec A$ means that σ is a prefix of the characteristic string of A (i.e., $\sigma = A(0)A(1)\cdots A(n)$ for some n).
4. Let $i \in \mathsf{N}$. If $\vec{x} \in \Sigma^\omega$ then $\vec{x}[i]$ is the ith symbol in $\vec{x}$. If $\vec{x} \in (\Sigma^k)^\omega$ and $1 \leq j \leq k$ then $\vec{x}[i,j]$ be the jth coordinate of $\vec{x}[i]$.
5. Let $\vec{x} \in (\Sigma^k)^\omega$ and $1 \leq j \leq k$. Then $\Pi_j(\vec{x})$ is the projection of $\vec{x}$ onto the jth (horizontal) coordinate. Formally $\Pi_j(\vec{x}) = \vec{x}[0,j]\vec{x}[1,j]\vec{x}[2,j]\cdots$. If $\mathcal{A} \subseteq (\Sigma^k)^\omega$ then $\Pi_i(\mathcal{A})$ is $\{\Pi_i(\vec{x}) : \vec{x} \in \mathcal{A}\}$.

Definition 5.2 A *Büchi Automaton* [2] is a nondeterministic finite automaton $\mathcal{A} = \langle Q, \Sigma, \Delta, s, F\rangle$ where Q is the set of *states*, Σ is the *alphabet*, Δ maps $Q \times \Sigma$ to 2^Q, $s \in Q$ (the *start state*), and $F \subseteq Q$, the *accepting states.* A Büchi Automaton will differ from a nondeterministic finite automaton in that we intend to run $\mathcal{A}$ on elements of Σ^ω. Let $\vec{x} \in \Sigma^\omega$. A *run of* $\mathcal{A}$ *on* $\vec{x}$ is a sequence $\vec{q} \in Q^\omega$ such that $\vec{q}(0) = s$ and $(\forall i)[\vec{q}[i+1] \in \Delta(\vec{q}[i], \vec{x}[i])]$. $\mathcal{A}$ *accepts* $\vec{x}$ if there exists a run $\vec{q}$ such that $(\exists^\infty i)[\vec{q}[i] \in F]$. $\mathcal{A}$ *accepts* $A \subseteq \Sigma^\omega$ if $(\forall x \in \Sigma^\omega)[x \in A$ iff $\mathcal{A}$ accepts x].

Definition 5.3 A subset of Σ^ω is *ω-regular* if there exists a Büchi automaton that accepts it. (There are many equivalent definitions of ω-regular sets [4, 13].)

Convention 5.4 We identify a Büchi automaton with the ω-regular set it accepts. In Definition 5.9 we will use this to formulate some very useful notation.

By representing sets of natural numbers via their characteristic sequences (which are elements of $\{0,1\}^\omega$) we can think of an ω-regular set over the alphabet $\Sigma = \{0,1\}$ as a subset of $\mathcal{P}(\mathsf{N})$. By representing a k-tuple of sets of natural numbers via an element of $(\{0,1\}^k)^\omega$ we can think of an ω-regular set over the alphabet $\Sigma = \{0,1\}^k$ as being a subset of $\mathcal{P}(\mathsf{N}) \times \cdots \times \mathcal{P}(\mathsf{N})$ (there are k copies of $\mathcal{P}(\mathsf{N})$). We denote the input to a Büchi automaton over the alphabet $\Sigma = \{0,1\}^k$ by $(A_1, \ldots, A_k)$ where A_i is the set whose characteristic string is the projection of the input onto the ith coordinate.

The following theorems from the literature link Büchi automata with formulas over $[S, <]^2$.

Proposition 5.5 ([1, 2, 4]) *Let $k_1, k_2 \in \mathsf{N}$. Let $\mathcal{C} \subseteq (\{0,1\}^{k_1+k_2})^\omega$. Assume that, for $1 \le i \le k_1$, for $A \in \mathcal{C}$, $\Pi_i(A) \in 0^*10^\omega$. The following are equivalent.*

1. *$\mathcal{C}$ is ω-regular.*
2. *There exists a formula $\phi(x_1, \ldots, x_{k_1}, X_1, \ldots, X_{k_2})$ over $[S, <]^2$ such that, using our convention that the x_i range over N while the X_i range over subsets of N,*

$$\mathcal{C} = \{(\{a_1\}, \ldots, \{a_{k_1}\}, A_1, \ldots, A_{k_2}) : \phi(a_1, \ldots, a_{k_1}, A_1, \ldots, A_{k_2})\}.$$

3. There are regular sets $U_1, \ldots, U_n, V_1, \ldots, V_n$ such that $\mathcal{C} = \bigcup_{i=1}^{n} U_i \cdot V_i^{\omega}$ and $(\forall i)[V_i^* = V_i]$. Note that if $\mathcal{C}$ is uncountable then we can assume without loss of generality that V_1^{ω} is uncountable.

Moreover these equivalences are constructive: given any of the objects { Büchi automaton, formula, the sets $\{U_i\}_{i=1}^{n}, \{V_i\}_{i=1}^{n}$ } one can effectively find the other two.

Definition 5.6 Let $\phi(X)$ be a formula over $[S,<]^2$. Then $AUT(\phi)$ is a Büchi automaton that recognizes $\{A : \phi(A)\}$. Such an automaton exists by Proposition 5.5. Note that it can be effectively found from ϕ.

Definition 5.7 Let $x \in \Sigma^{\omega}$ and $a \in \Sigma$. The expression $rm(a,x)$ denotes the string that results from removing every occurrence of a from x. Note that $rm(a,x)$ could be finite, hence $rm(a,x) \in (\Sigma - \{a\})^{\omega} \cup (\Sigma - \{a\})^*$. Let $\mathcal{B} \subseteq \Sigma^{\omega}$. $rm(a,\mathcal{B}) = \{x \in (\Sigma - a)^{\omega} : (\exists y \in \mathcal{B})[rm(a,y) = x]\}$. Since we identify a Büchi automaton $\mathcal{A}$ with the set $\{A : \mathcal{A} \text{ accepts } A\} \subseteq \Sigma^{\omega}$ the notation $rm(a,\mathcal{A})$ make sense.

Proposition 5.8 *If $1 \le i \le k$, $\mathcal{A}, \mathcal{B}$ are ω-regular subsets of $(\{0,1\}^k)^{\omega}$, $\sigma \in \{0,1\}^k$, then $\Pi_i(\mathcal{A})$, $rm(\sigma,\mathcal{A})$, $\mathcal{A} \cup \mathcal{B}$, $\mathcal{A} \cap \mathcal{B}$, and $\overline{\mathcal{A}}$ are ω-regular. One can obtain the automaton for $\mathcal{A} \cap \mathcal{B}$ (and the others) effectively by using a cross-product construction. (Büchi [2] showed that the class of ω-regular sets is closed under union, intersection, and complementation. See [4] for an exposition and see [16] for a more efficient proof.)*

Definition 5.9 We identify a Büchi automaton $\mathcal{A}$ with both the ω-regular language that it accepts and the query over $[S,<]^2$ that it is equivalent to by Proposition 5.5. Let $\mathcal{A}$ be a Büchi automaton.

1. If $\mathcal{A}$ uses alphabet $\{0,1\}$ and $A \subseteq \mathsf{N}$ then $\mathcal{A}(A)$ will mean that $\mathcal{A}$ accepts the characteristic string of A.

2. $\mathcal{A}$ is *uncountable* if the corresponding ω-regular set is uncountable.

3. If $\mathcal{A}$ and $\mathcal{B}$ are Büchi automaton over $\{0,1\}$ then $\mathcal{A}\cap\mathcal{B}$ is the automaton that accepts only those elements of $\{0,1\}^\omega$ accepted by both $\mathcal{A}$ and $\mathcal{B}$. This automaton exists by Proposition 5.8.

4. If $\phi(X)$ is a query over $[S,<]^2$ and $\mathcal{A}$ uses alphabet $\{0,1\}$ then $\mathcal{A}\cap\phi(X)$ is the Büchi automaton that accepts only those elements of $\{0,1\}^\omega$ that are both accepted by $\mathcal{A}$ and satisfy ϕ. This automaton exists by Propositions 5.5 and 5.8. We may use the notation $\mathcal{A}\cap\phi$ at times.

5. Let σ be a finite string. If there is a run of $\mathcal{A}$ on σ that contains an accepting state n times then σ is *n-accepted by $\mathcal{A}$*. Note that if $\mathcal{A}\cap\mathcal{B}$ n-accepts σ then both $\mathcal{A}$ n-accepts σ and $\mathcal{B}$ n-accepts σ.

Proposition 5.10 ([11]) *Let $\mathcal{A}$ be a Büchi automaton, $n \in \mathsf{N}$, and $\sigma \in \Sigma^*$ be such that $\{A : (\mathcal{A}$ accepts $A) \wedge (\sigma \prec A)\}$ is uncountable. Then there exists σ' such that*

1. *$\sigma \prec \sigma'$,*

2. *$\{A : (\mathcal{A}$ accepts $A) \wedge (\sigma' \prec A)\}$ is uncountable, and*

3. *$\mathcal{A}$ n-accepts σ'.*

The string σ' can be found from $(\sigma, \mathcal{A}, n)$ in a uniform effective fashion.

Definition 5.11 Let EXT be a partial computable function that takes a Büchi automaton $\mathcal{A}$, a string σ, and the number $n = |\sigma|$, and returns the σ' from Proposition 5.10.

Proposition 5.12 *Let $\{\mathcal{A}_s\}_{s=1}^\infty$ be a sequence of Büchi automata and $\{\sigma_s\}_{s=1}^\infty$ be elements of Σ^* such that the following are true.*

1. *$\mathcal{A}_1 \supseteq \mathcal{A}_2 \supseteq \mathcal{A}_3 \cdots$.*

2. *$\sigma_1 \prec \sigma_2 \prec \sigma_3 \prec \cdots$.*

3. *There exists an increasing sequence $n_1 < n_2 < n_3 \cdots$ such that for all s, $\mathcal{A}_s$ n_s-accepts σ_s.*

Then the string $A = \lim_{s\to\infty} \sigma_s \in \Sigma^\omega$ is accepted by all $\mathcal{A}_s$. In addition, if the sequence $\sigma_1, \sigma_2, \ldots$ is computable then A is computable (this is obvious).

Proof: Fix s. We show that A is accepted by $\mathcal{A}_s$. For any $t \geq s$, $\mathcal{A}_s - \mathcal{A}_t$ is ω-regular by Proposition 5.8. Hence $\mathcal{A}_t$ can be built by intersecting $\mathcal{A}_s$ with a Büchi automaton. Hence, any finite string that $\mathcal{A}_t$ n-accepts is also n-accepted by $\mathcal{A}_s$. Therefore σ_t is n_t-accepted by $\mathcal{A}_s$. By an easy application of König's infinite lemma one can construct a run of A that shows $\mathcal{A}_s$ accepts A. ▮

Proposition 5.13 (Lemma 16 of [11]) *If V is a regular set such that $|V^\omega| \geq 2$ then V^* contains two distinct strings of the same length.*

Lemma 5.14 *Let $a \in \mathsf{N}$. There is a partial computable function that, given an uncountable Büchi automaton $\mathcal{A}$, returns $n \in \mathsf{N}$ such that*

$$(\forall b \in \{0,1\})[(\mathcal{A} \cap (X(n) = b)) \textit{ is uncountable}].$$

(See Definition 5.9 for the definition of $\mathcal{A} \cap \phi$.)

Proof: Let $\mathcal{A}$ be uncountable. By Propositions 5.5 and 5.13, applied to $\mathcal{A}$, we can find DFA's for regular sets U and V such that the following hold.

1. $A \in U \cdot V^\omega \Rightarrow \mathcal{A}(A)$.
2. $V = V^*$.
3. V^ω is uncountable.
4. There exists $v_0, v_1 \in V$ and $i \in \mathsf{N}$ such that $|v_0| = |v_1|$, $v_0[i] = 0$, and $v_1[i] = 1$. (v_0, v_1, and i can be found by searching.)

Fix $u \in U$. Let

$$\begin{aligned} |u| =&\ r \text{ (so } u \text{ determines membership for } \{0, \ldots, r-1\}) \\ |v_0| =&\ |v_1| = s \text{ (so } 0 \leq i \leq s-1) \\ n =&\ r + i \end{aligned}$$

Let $b =\in \{0,1\}$. If $A \in uv_0 \cdot V^\omega$ then (1) $A \in U \cdot V^\omega$, so $\mathcal{A}(A)$, and (2) $uv_0 \prec A$ so $A(n) = b$. Since V^ω is uncountable we have that $(\mathcal{A} \cap X(n) = b)$ is uncountable. ▮

Definition 5.15 $FIND$ is the partial computable function that was shown to exist in Lemma 5.14, i.e., the n in the proof of Lemma 5.14.

Lemma 5.16 *There is a computable function $RESTRICT$ that behaves as follows.*

1. *The input to $RESTRICT$ is an ordered pair $(\mathcal{A}, \phi)$ where $\mathcal{A}$ is a Büchi automaton and ϕ is a query over $[S, <]^2$.*
2. *$RESTRICT(\mathcal{A}, \phi)$ is a Büchi automaton.*
3. *$RESTRICT(\mathcal{A}, \phi)$ accepts either $\mathcal{A} \cap \phi$ or $\mathcal{A} \cap \neg\phi$. We abbreviate this by $RESTRICT(\mathcal{A}, \phi) = \mathcal{A} \cap \phi$ or $RESTRICT(\mathcal{A}, \phi) = \mathcal{A} \cap \neg\phi$. (See Definition 5.9 for the definition of $\mathcal{A} \cap \phi$.)*
4. *If $\mathcal{A}$ is uncountable then $RESTRICT(\mathcal{A}, \phi)$ is uncountable.*

(In essence $RESTRICT$ picks out which of ϕ or $\neg\phi$ has an uncountable intersection with $\mathcal{A}$ and then produces the automaton that accepts that intersection.)
Proof: $RESTRICT(\mathcal{A}, \phi)$ is computed as follows. By Proposition 5.5, Proposition 5.8, (of this paper) and Lemma 21 of [11] one can effectively construct Büchi automaton for each of $\mathcal{A} \cap \phi$ and $\mathcal{A} \cap \neg\phi$ and determine, for each one, if it is uncountable. If $\mathcal{A} \cap \phi$ is uncountable then output the automaton for it, else output the automaton for $\mathcal{A} \cap \neg\phi$. Clauses 1,2, and 3 are clearly satisfied. If $\mathcal{A}$ is uncountable then one of $\{\mathcal{A} \cap \phi, \mathcal{A} \cap \neg\phi\}$ is uncountable, hence clause 4 is satisfied. ▮

5.1.2 $COMP \notin QBC[L]$

The proof of the following theorem is similar to the proof that $COMP \notin QEX[S, <]$ from [11] (in that paper it was called $REC \notin QEX[S, <]$). We include the proof for completeness and because we are going to discuss variants of it.

Theorem 5.17 $COMP \notin QBC[S, <]^2$.

Proof: Given a QIM M that asks queries in $[S,<]^2$ we try to construct a computable set A that is not QBC-inferred by M. By Convention 2.13 the ith query M asks is ϕ_i.

We use the functions AUT (from Definition 5.6), EXT (from Definition 5.11), and $FIND$ (from Definition 5.15) in the construction. We will need $FIND$ because when we diagonalize we want either choice (using 0 or 1) to still leave an uncountable number of functions.

We use the function $RESTRICT$ (from Lemma 5.16) in the proof that the construction works. We will also use the computable function BIT defined as follows.

$$BIT(\mathcal{A},\phi) = \begin{cases} 1 & \text{if } RESTRICT(\mathcal{A},\phi) = \mathcal{A} \cap \phi; \\ 0 & \text{if } RESTRICT(\mathcal{A},\phi) = \mathcal{A} \cap \neg\phi. \end{cases}$$

We construct A in stages. At every stage s we will have the following.

1. $t_s \in \mathsf{N}$ and $\vec{c}_s \in \{0,1\}^{t_s}$. $\vec{c}_s$ represents answers to the first t_s queries over L, which will be $\phi_1, \ldots, \phi_{t_s}$ by our convention.

2. An uncountable Büchi automaton $\mathcal{A}_s$ which captures $\vec{c}_s$ (see Definition 2.14). It will be used to make the set A satisfy the query answers given.

3. $\sigma_s \in \{0,1\}^*$. $\mathcal{A}_s(X)$ implies $\sigma_s \preceq X$, so σ_s is an initial segment of A.

Either every stage of this construction terminates or the very failure of a stage to terminate allows us to construct a computable set B that is not QBC-inferred by M.

CONSTRUCTION

Stage 0: $t_0 = 0$, $\vec{c}_0 = \lambda$, and $\mathcal{A}_0$ accepts $\{0,1\}^\omega$, and $\sigma_0 = \lambda$.

Stage $s+1$*:* We can inductively assume that $\mathcal{A}_s$ is uncountable and captures $\vec{c}_s$. Dovetail the following procedure for all $(t,\vec{c}) \in \mathsf{N} \times \{0,1\}^*$ such that $t \geq 1$ and $\vec{c} \in \{0,1\}^t$. If any of them *terminate with success* then its actions determine how the construction should proceed. No parameter is changed until this happens.

1. We will simulate what happens if t more queries (namely $\phi_{t_s+1}, \ldots, \phi_{t_s+t}$) are asked and ϕ_{t_s+i} is answered with $\vec{c}[i]$. Let $\mathcal{A}^0 = \mathcal{A}_s$. For $1 \leq i \leq t$ let
$$\mathcal{A}^i = \begin{cases} \mathcal{A}^{i-1} \cap AUT(\phi_{t_s+i}) & \text{if } \vec{c}[i] = 1; \\ \mathcal{A}^{i-1} \cap AUT(\neg\phi_{t_s+i}) & \text{if } \vec{c}[i] = 0. \end{cases}$$
Let $\mathcal{A} = \mathcal{A}^t$. If $\mathcal{A}$ is countable or finite then *terminate with failure*.

2. ($\mathcal{A}$ is uncountable and captures $\vec{c}_s\vec{c}$.) Let $e = M(\vec{c}_s\vec{c})$, the conjecture that M makes upon getting answers $\vec{c}_s\vec{c}$. Hence we try to diagonalize against φ_e.

3. Let $n = FIND(\mathcal{A})$. Run $\varphi_e(n)$.

4. If $\varphi_e(n)\downarrow$ then we can *terminate with success*. Proceed as follows.
$$\begin{aligned} b &= (1 - \varphi_e(n)) \\ \mathcal{A}_{s+1} &= \mathcal{A} \cap (X(n) = b) \\ \sigma_{s+1} &= EXT(\sigma_s, \mathcal{A}_{s+1}) \\ t_{s+1} &= t_s + t \\ \vec{c}_{s+1} &= \vec{c}_s\vec{c}. \end{aligned}$$
We have caused φ_e and A to disagree on n, $\mathcal{A}_{s+1}$ to be uncountable, and $\mathcal{A}_{s+1}$ to capture $\vec{c}_{s+1}$. (Note that if step 4 is never reached then either some other choice of $(t, \vec{c})$ terminated with success first or φ_e fails to converge on n.)

END OF CONSTRUCTION

There are two cases.

Case 1: Every stage terminates. The sequences $\{\mathcal{A}_s\}_{s=1}^{\infty}$ and $\{\sigma_s\}_{s=1}^{\infty}$ satisfy the premises of Proposition 5.12. Let A be the resulting computable set. Imagine M trying to infer A. For all s note that the vector of answers to the first t_s queries is $\vec{c}_s$. Let e_s be the guess made after t_s queries are made and answered. During stage $s+1$ program φ_{e_s} is diagonalized against. Hence φ_{e_s} does not decide an A, so M does not QBC-infer A.

Case 2: There exists a stage $s+1$ that does not terminate. Hence for *any* choice of $(t, \vec{c})$ the resulting program φ_e fails to converge on some point. We use this to effectively construct $\{\mathcal{B}_s\}_{s=1}^{\infty}$ and $\{\tau_s\}_{s=1}^{\infty}$ that satisfy Proposition 5.12.

```
$\vec{d_0} = \vec{c_s}$
$\mathcal{B}_0 = \mathcal{A}_s$
For $i = 1$ to $\omega$
  $\vec{d_i} = \vec{d}_{i-1} \cdot BIT(\mathcal{B}_{i-1}, \phi_{t_s+i})$
  $\mathcal{B}_i = RESTRICT(\mathcal{B}_{i-1}, \phi_{t_s+i})$
        (* $\phi_{t_s+i}$ is $(t_s + i)^{\text{th}}$ query *)
  $\tau_i = EXT(\tau_{i-1}, \mathcal{B}_i)$
```

Clearly $\{\mathcal{B}_t\}_{t=1}^{\infty}$ and $\{\tau_i\}_{i=1}^{\infty}$ satisfy the premise of Proposition 5.12. Let B be the resulting computable set. We show that M does not QBC-infer A. Let $\vec{d}$ be the limit of $\vec{d_i}$. Imagine M trying to infer B. For all $t' = t + t_s$ let e_t be the guess made by M after t' queries are made and answered. Let $\vec{d}[1..t'] = \vec{c_s} \cdot \vec{c}$. During stage $s+1$ $(t, \vec{c})$ was considered. By the construction of $\mathcal{B}_i$ we know that the resulting automaton $\mathcal{A}$ is uncountable. Hence step 3 (of stage $s + 1$ of the main construction) was reached. Note that $e_t = \varphi_{M(\vec{c}[t])}$. Since stage $s + 1$ never terminated there is a point n such that $\varphi_{e_t}(n)\uparrow$. Hence φ_{e_t} is not an index for B. Since this reasoning holds for any $t > t_s$, M does not QBC-infer B ▮

Theorem 5.18 *If $L \leq_{\mathsf{N}} [S, <]^2$ then $COMP \notin QBC[L]$.*

Proof: By Theorem 5.17 $COMP \notin QBC[S, <]$. By the premise $L \leq_{\mathsf{N}} [S, <]^2$. Hence, by Theorem 4.4.3 $COMP \notin QBC[L]$. ▮

5.2 $L \leq_{\mathsf{N}} [S, <]^2 \Rightarrow COMP \notin [1, d]QBC^a[L]$

We first show that, for all d, $COMP \notin [1, d]QBC[S, <]^2$. We then use Lemma 2.7 to obtain, for all a, d, $COMP \notin [1, d]QBC^a[S, <]^2$. We then use Theorem 4.4.3 to obtain, for all a, d, for all $L \leq_{\mathsf{N}} [S, <]^2$, $COMP \notin [1, d]QBC^a[L]$.

Theorem 5.19 *$COMP \notin [1, d]QBC[S, <]^2$.*

Proof: We use the same conventions as in Theorem 5.17. $RESTRICT$, BIT, and $FIND$ are as in Theorem 5.17.

Fix d. Given $\{M_1, \ldots, M_d\}$, a team of d QIMs that asks queries in L, we try to construct a computable set A that is not QBC-inferred by any M_m. By Convention 2.13, for $1 \leq m \leq d$, the ith query that M_m makes is ϕ_i.

The construction will proceed in stages. At every stage s we will have $t_s \in \mathsf{N}$, $\vec{c}_s \in \{0,1\}^{t_s}$, an uncountable Büchi Automata $\mathcal{A}_s$, σ_s, and $POSS_s \subseteq \{1,\ldots,m\}$. The parameters $\vec{c}_s$, $\mathcal{A}_s$, and σ_s are similar to their counterparts in Theorem 5.17. $POSS_s$ represents the set of machines that might infer A.

The construction may fail to terminate. If this occurs then there exists m, σ and $\mathcal{A}$ such that *any* set with initial segment σ that is accepted by $\mathcal{A}$ is not inferred by M_m. We will then (noncontructively) restart our construction such that (1) we no longer care about M_m and (2) the set we construct has initial segment σ and is accepted by $\mathcal{A}$. The construction may fail again, in which case we can eliminate another machine and restart again.

If less than m stages of the construction fail to terminate then the construction will produce a computable set A that is not BC inferred by any M_m. The algorithm for A will need some finite information about which stages did not terminate along with the code for the construction. If m stages of the construction fail to terminate then these m failures will allow us to construct a computable set B that is not QBC-inferred by M.

CONSTRUCTION

Stage 0: $t_0 = 0$, $\vec{c}_0 = \lambda$, $\mathcal{A}_0$ accepts $\{0,1\}^\omega$, $\sigma_0 = \lambda$, and $POSS_0 = \{1,\ldots,d\}$.

Stage $s+1$: We can inductively assume that $\mathcal{A}_s$ is uncountable and captures $\vec{c}_s$. Let m be such that $1 \le m \le d$ and $s \equiv m \pmod{d}$. If $m \notin POSS_s$ then go to the next stage. If $m \in POSS_s$ then we try to diagonalize against M_m. Dovetail the procedure described in Theorem 5.17 for all $(t,\vec{c}) \in \mathsf{N} \times \{0,1\}^*$ such that $t \ge 1$ and $\vec{c} \in \{0,1\}^t$, except that we use M_m instead of M. If any of them *terminate with success* then take the same action as in Theorem 5.17. If none of the $(t,\vec{c})$ cause a termination with success then (nonconstructively) let $POSS_{s+1} = POSS_s - \{m\}$.

END OF CONSTRUCTION

Case 1: For all s, $POSS_s \neq \emptyset$. The parameters $\{\mathcal{A}_s\}_{s=1}^\infty$ and $\{\sigma_s\}_{s=1}^\infty$ satisfy the premises of Proposition 5.12. Let A be the resulting set. The sequence $\{\sigma_s\}_{s=1}^\infty$ is computable, but the algorithm for it needs finite information about which stages did not terminate. Even so, the set A is computable. We show that, for all m, M_m does not QBC-infer A. There are two cases.

1. $m \in \lim_{s\to\infty} POSS_s$. A cannot be QBC-inferred by M_m by reasoning similar to that of Case 1 of Theorem 5.17 specialized to the stages $s \equiv m \pmod{d}$.

2. $m \notin \lim_{s\to\infty} POSS_s$. Let s be such that $m \in POSS_s - POSS_{s+1}$. Imagine M_m trying to infer A. For all $t' = t + t_s$ let e_t be the guess made by M after t' queries are made and answered. Let $\vec{c}$ be the correct set of answers to the first t queries and let $\vec{c} = \vec{c}_s \cdot \vec{c}$. The rest of this proof is similar to Case 2 of Theorem 5.17.

Case 2: There exists s such that $POSS_s = \emptyset$. A new computable set B can be constructed that is not BC inferred by the team. This can be proven with reasoning similar to Case 2 of Theorem 5.17. ▮

Corollary 5.20 *If $a, d \in \mathsf{N}$ then $COMP \notin [1,d]QBC^a[S,<]^2$.*

Proof: By Theorem 5.19, with parameter $d(a+1)$, we have $COMP \notin [1, d(a+1)]QBC[S,<]^2$. Lemma 2.7 yields $COMP \notin [1,d]QBC^a[S,<]^2$. ▮

Corollary 5.21 *If $a, d \in \mathsf{N}$ and $L \leq_{\mathsf{N}} [S,<]^2$ then $COMP \notin [1,d]QBC[L]$.*

Proof: By Corollary 5.20 $COMP \notin [1,d]QBC^a[S,<]$. By the premise $L \leq_{\mathsf{N}} [S,<]^2$. Hence, by Theorem 4.4.3 $COMP \notin [1,d]QBC^a[L]$. ▮

5.3 $L \leq_{\mathsf{N}} [S,<]^2 \Rightarrow PEX_{n+1} - QEX_n^a[L] \neq \emptyset$

We first obtain $PEX_{n+1} - QEX_n[S,<]^2 \neq \emptyset$ by showing that $[S,<]^2$ is nice.

Lemma 5.22 *$[S,<]^2$ is nice.*

Proof:

Let W be the computable set whose characteristic sequence is $0^1 1 0^2 1 0^3 1 \cdots$. We show that $[S,<]^2$ is $(\mathsf{N}, W, 2)$-nice.

Let $e \in \mathsf{N}$. Assume

$$(\exists W' \subseteq \mathsf{N})[\psi(W') \wedge (W' =^{=2e} W)].$$

We show that

$$(\exists W'' \subseteq \mathsf{N})[\psi(W'') \wedge (W'' =^{=2e+2} W)].$$

(See Definition 2.10 for the definition of $=^{=a}$.)

Let the characteristic sequence of W' be $0^{m_1}10^{m_2}10^{m_3}\cdots$. Note that $(\forall^\infty i)[m_i = i]$.

By Proposition 5.5 there exists regular sets U and V such that $0^{m_1}10^{m_2}\cdots \in U \cdot V^\omega$ and $V = V^*$. Let the DFA for V be $\langle Q, \{0,1\}, \delta, s, F\rangle$. Note that $\delta : Q \times \{0,1\} \rightarrow Q$. Let $\overline{\delta}$ be the natural extension of δ to domain $Q \times \{0,1\}^*$.

Our plan is to represent $0^{m_1}10^{m_2}10^{m_3}1\cdots$ as $uv_1v_2v_3\cdots$ where $u \in U$ and $v_i \in V$ and then to replace one of the v_i with $v_i' \in V$ such that $v_i =^{=2} v_i'$. In order to find such a v_i' we will need x, y, z_1, z_2 such that $v_i = z_1xz_2$, $v_i' = z_1yz_2$, $|x| = |y|$, and the DFA cannot tell x and y apart.

We define an equivalence relation on $\{0,1\}^*$. Associate to every $x \in \{0,1\}^*$ the set

$$INFO(x) = \{(q_1, q_2) : \overline{\delta}(q_1, x) = q_2\}.$$

Define $x \sim y$ iff $INFO(x) = INFO(y)$. Note that there are $\le |Q|^2$ equivalence classes.

We assume that $x \sim y$ and $z_1xz_2 \in V$ and show that $z_1yz_2 \in V$. Let q_1 and q_2 be such that $\overline{\delta}(s, z_1) = q_1$ and $\overline{\delta}(q_1, x) = q_2$. Since $z_1xz_2 \in V$ we have $\overline{\delta}(q_2, z_2) \in F$. Clearly $(q_1, q_2) \in INFO(x) = INFO(y)$. Since $\overline{\delta}(s, z_1) = q_1$, $\overline{\delta}(q_1, y) = q_2$, and $\overline{\delta}(q_2, z_2) \in F$ we have $z_1yz_2 \in V$.

The number of equivalence classes is $\le |Q|^2$. Let $n > |Q|^2$. Consider the n strings $\{0^i10^j : i + j = n\}$. Two of these must be equivalent, say $0^{i_1}10^{j_1}$ and $0^{i_2}10^{j_2}$. We intend to replace some occurrence of the substring $0^{i_1}10^{j_1}$ in the characteristic string of W' with $0^{i_2}10^{j_2}$. To ensure the result is still in $U \cdot V^\omega$ this must be done carefully.

Since $(\forall^\infty i)[m_i = i]$ the string 0^n10^n is a substring of W' infinitely often. Since $W' \in U \cdot V^\omega$, $W =^{=2e} W'$, and $V = V^*$ there exists $u \in U$, $v_1, v_2, \ldots \in V$, and $i_0 \in \mathsf{N}$ such that the following happens.

1. W' has characteristic string $uv_1v_2v_3\cdots$.
2. $(\forall j \ge |uv_1v_2\cdots v_{i_0}|)[W(j) = W'(j)]$.
3. $(\exists i > i_0)(\exists z_1', z_2')[v_i = z_1'0^n10^nz_2']$.

Since $n \ge i_1, j_1$ we can rewrite v_i as $z_10^{i_1}10^{j_1}z_2$. Since $0^{i_1}10^{j_1} \sim 0^{i_2}10^{j_2}$ and $v_i \in V$ we have $v_i' = z_10^{i_2}10^{j_2}z_2 \in V$. Let W'' be the set whose characteristic string is $uv_1v_2v_3\cdots v_{i-1}v_i'v_{i+1}v_{i+2}\cdots$.

Since $v_i' \in V$ the characteristic string of W'' is in $U \cdot V^\omega$. Hence $\psi(W'')$ holds.

Since $|v_i| = |v'_i|$ and $v_i =^{=2} v'_i$ we have $W' =^{=2} W''$. Since $i > i_0$ the two places where W'' and W' disagree are also places where W'' and W disagree. Since $W =^{=2e} W'$ we have $W =^{=2e+2} W''$. ▮

Theorem 5.23 *If $L \leq_{\mathsf{N}} [S,<]^2$ then*

1. $PEX_{n+1} - QEX^a_n[L] \neq \emptyset$.
2. $QEX^a_n[L] \subset QEX^a_{n+1}[L]$. *(This is an easy consequence of part 1.)*

Proof: By Lemma 5.22 $[S,<]^2$ is nice. By Theorem 4.8.2 L is nice. By Theorem 3.2 $PEX_{n+1} - QEX^a_n[L] \neq \emptyset$. ▮

5.4 $L \leq_{wb} [S,<]^2 \Rightarrow PEX_{n+1} - QEX^a_n[L] \neq \emptyset$

We need a slightly stronger version of Lemma 5.22, in what follows.

Definition 5.24 Let $E \subseteq \mathsf{N}$, E infinite, $E = \{e_1 < e_2 < e_3 \cdots\}$. Let $W \subseteq \mathsf{N}$. The set E *restricted to* W is $\{e_i : i \in W\}$. We denote this by $E \upharpoonright W$. Intuitively $E \upharpoonright W$ is obtained by using the characteristic sequence of W to determine which *members of* E to put into our set. Note that $E \subseteq E \upharpoonright W$.

Lemma 5.25 *Let W be as in Lemma 5.22. Let E be such that the singleton set $\{E\}$ is ω-regular. Then $[S,<]^2$ is $(E, E \upharpoonright W, 2)$-nice.*

Proof: The definition of $(E, E \upharpoonright W, 2)$-nice deals with subsets of E. Rather than deal with such subsets directly, we will deal with sets $W \subseteq \mathsf{N}$ and look at $E \upharpoonright W$.

Let $e \in \mathsf{N}$. Assume

$$(\exists W' \subseteq \mathsf{N})[\psi(E \upharpoonright W') \wedge (E \upharpoonright W' =^{2e} E \upharpoonright W)].$$

We show that

$$(\exists W'' \subseteq \mathsf{N})[\psi(E \upharpoonright W'') \wedge (E \upharpoonright W'' =^{=2e+2} E \upharpoonright W)].$$

Let the characteristic sequence of W' be $0^{m_1}10^{m_2}10^{m_3}\cdots$. Note that

$$(\forall n)(\exists^\infty i)[(m_i, m_{i+1}) = (n, n)].$$

Let $\mathcal{A} = \{A : \Pi_1(A) = E \wedge \psi(\Pi_2(A))\}$. Since $\{E\}$ is ω-regular $\mathcal{A}$ is ω-regular (use Proposition 5.8). Since $\mathcal{A}$ is ω-regular $\mathcal{B} = \Pi_2(rm(00, \mathcal{A}))$ is ω-regular (use Propositions 5.8). Note that $W' \in \mathcal{B}$.

We now find a set W'' in a manner identical to that of Lemma 5.22. Formally we end up with $E \restriction W''$ as our desired set. ▮

Theorem 5.26 *If $L \leq_{wb} [S, <]^2$ then the following are true.*

1. $PEX_{n+1} - QEX_n^a[L] \neq \emptyset$.
2. $QEX_n[L] \subset QEX_{n+1}[L]$. *(This is an easy consequence of part 1.)*

Proof: Let $(P, \{E_d^\$\}_{d=1}^\infty, \{f\}_{d=1}^\infty)$ be the well behaved parameters. We show that the conditions of Theorem 4.8.2 hold with L as above, $L^\$ = [S, <]^2$, W from Lemma 5.22, $W_d^\$ = E_d^\$ \restriction W$.

Fix d. Clearly $W_d^\$$ is a computable subset of $E_d^\$$. Since the parameters are well behaved the singleton set $\{E_d^\$\}$ is ω-regular. Hence, by Lemma 5.25, $[S, <]^2$ is $(E_d^\$, W_d^\$, 2)$-nice. By definition $f_d^{-1}(W_d^\$) = W$. These are all the conditions needed.

By Theorem 4.8.2, L is $(P, W, 2)$-nice. Since P and W are computable, L is nice. By Theorem 3.2, $PEX_{n+1} - QEX_n^a[L] \neq \emptyset$. ▮

5.5 $L \leq_{wb} [S, <]^2 \Rightarrow L$ is QCD

Theorem 5.27 *If $L \leq_{wb} [S, <]^2$ then L is QCD.*

Proof: Assume $L \leq_{wb} [S, <]^2$ with parameters $(E, \{E_d^\$\}_{d=1}^\infty, \{f_d\}_{d=1}^\infty)$. We show that $[S, <]^2$ is uniformly $E_d^\$$-CD.
ALGORITHM

1. Input(ψ, d) (ψ is a query over $[S, <]^2$ and $d \geq 1$.)
2. Find the query ψ' (over $[S, <]^2$) that accepts $\{A : A \subseteq E_d^\$$ and $\psi(A)\}$. This is possible since $\{E_d^\$\}_{d=1}^\infty$ is well behaved (also use Proposition 5.5).

3. Test if ψ' is uncountable. Note that ψ' is uncountable iff ψ' is $E_d^{\$}$-uncountable. Output the answer.

END OF ALGORITHM

Since $[S,<]^2$ is uniformly $E_d^{\$}$-$CD$, by Theorem 4.8.4 L is QCD. ∎

Theorem 5.28 *If $L \le_{\mathsf{N}} [S,<]^2$ then L is QCD.*

Proof: If $L \le_{\mathsf{N}} [S,<]^2$ then clearly $L \le_{wb} [S,<]^2$. Apply Theorem 5.27 ∎

6 $[+,<] \le_{\mathsf{N}} [S,<]^2$

We show that $[+,<] \le_{\mathsf{N}} [S,<]^2$. We will use some ideas from the proof that first order Presburger arithmetic is decidable by a reduction to the weak second order theory with successor (see [15]).

Lemma 6.1 $[+,<] \le_{\mathsf{N}} [S,<]^2$ *with parameters* (POW_2, f) *where f is the function that maps 2^i to i.*

Proof:

Let $\phi(X)$ be a query over $[+,<]$. Our only concern is when $X \subseteq \mathrm{POW}_2$. We intend to replace X by a variable $\mathcal{X}$ such that as X ranges over POW_2, $\mathcal{X}$ ranges over N. The statement $2^i \in X$ will be coded by $i \in \mathcal{X}$.

We will replace the original first order variables with second order variables. These second order variables will be constrained to be finite sets. We will code numbers into finite sets by the following convention.

Definition 6.2 Let Y be a finite set. Let $Y = \{a_1, \ldots, a_k\}$. The *number coded by* Y is $\sum_{j=1}^{k} 2^{a_j}$. We denote this number by *code*(Y). Note that *code* is a bijection from finite subsets of N to N.

Example 6.3 The set $\{3,5\}$ represents the number $2^3 + 2^5 = 40$. We will think of it as the string which has a 1 in the 3rd and 5th place, namely 10100.

We will then be able to replace $+$ by S and quantifiers as explained below. At that point we will have a query in $[S,<]^2$.

Definition 6.4 Let Y,Z be second order variables. $FIN(Y)$ is the formula that states that Y is finite, namely

$$(\exists x_1)(\forall x_2)[x_2 \geq x_1 \Rightarrow x_2 \notin Y].$$

$ONE(Y,Z)$ is the formula that states $|Y| = 1$ and the one element in Y is also in Z, namely

$$(\exists x_1)[(x_1 \in Y) \wedge (x_1 \in Z) \wedge (\forall x_2 \neq x_1)[x_2 \notin Y]].$$

The following reduction transforms a formula ϕ of $[+,<]$ into a formula of $[S,<]^2$.
REDUCTION OF $[+,<]$ TO $[S,<]^2$, abbreviated RED

1. Input(ϕ).
2. Replace all occurrence of $s < t$ with $(\exists x)[x \neq 0 \wedge s + x = t]$. Hence the formula has no $<$ in it. Call the resulting formula ϕ.
3. If $\phi = \phi_1 \wedge \phi_2$ then output RED(ϕ_1) $\wedge$ RED(ϕ_2). If $\phi = \neg\phi_1$ then output $\neg$RED(ϕ_1). If $\phi = (\exists y)[\phi_1(y)]$ then output $(\exists Y)[FIN(Y) \wedge$ RED$(\phi_1(y))]$. (Make sure that when y becomes a second order variable that variable is Y.) For the rest of this algorithm assume ϕ is atomic.
4. ϕ is $n \in X$ for some natural number n. If $n \notin$ POW$_2$ then replace this with $1 \neq 1$. If $n = 2^i$ then replace with $i \in \mathcal{X}$.
5. ϕ is $y = n$ for some natural number n. Replace with $Y = code^{-1}(n)$.
6. ϕ is $y \in X$. Replace this with $ONE(Y, \mathcal{X})$. We are saying that Y is a set that codes a number, Y is a singleton set (hence it codes a power of 2, say 2^i, by having $i \in Y$), and that number is in X (coded by $i \in \mathcal{X}$).
7. ϕ is $y_1 + y_2 = y_3$. We will replace y_1, y_2, y_3 with second order variables (if they ever get quantified over then they will be forced to be finite). We will obtain a formula that codes $code(Y_1) + code(Y_2) = code(Y_3)$. Let C be a new second order variable. C will represent bit carries.

Let *place* and *carry* be the functions from $\{0,1\}^3$ to $\{0,1\}^1$ such that for $b_1, b_2, b_3 \in \{0,1\}$ the sum $b_1 + b_2 + b_3$, when written in base 2, is $carry(b_1, b_2, b_3) \cdot place(b_1, b_2, b_3)$. (Note that here and elsewhere $\cdot$ means concatenation, not multiplication.)

Replace $y_1 + y_2 = y_3$ by the formula that begins with $(\exists C)(\forall i)$ and then has the conjunction of the following.

(a) $C(0) = 0$.

(b) For every $(b_1, b_2, b_3) \in \{0,1\}^3$ the formula

$$[(Y_1[i] = b_1) \wedge (Y_2[i] = b_2) \wedge (C[i] = b_3)] \Rightarrow$$
$$[(Y_3[i] = place(b_1, b_2, b_3)) \wedge (C[i+1] = carry(b_1, b_2, b_3))].$$

8. Output the formula obtained.

END OF REDUCTION

We need to show that the above algorithm is a reduction. The domain condition is clearly satisfied. We show the equivalence condition.

A formula $\phi(x_1, \ldots, x_n, X)$ will map to a formula $\phi^{\$}(X_1, \ldots, X_n, X)$. One can show by induction on the formation of a formula that for any $a_1, \ldots, a_n \in \mathsf{N}$ and $A \subseteq \mathrm{POW}_2$

$$\phi(a_1, \ldots, a_n, A) \text{ iff } \phi^{\$}(code^{-1}(a_1), \ldots, code^{-1}(a_n), f(A)).$$

Hence for queries $\phi(X)$ we obtain

$$(\forall A \subseteq \mathrm{POW}_2)[\phi(A) \text{ iff } \phi^{\$}(f(A))].$$

Hence the equivalence condition is satisfied. ▮

Lemma 6.5 $[+, <, \mathrm{POW}_2] \leq_{\mathsf{N}} [S, <]^2$ *with parameters* $(\mathrm{POW}_2, \mathsf{N}, f)$ *where* f *maps* 2^i *to* i.

Proof: We modify the proof of Lemma 6.1 by saying where to map the new atomic formulas that may occur. The formula $\mathrm{POW}_2(x)$ maps to $|X| = 1$. ▮

7 $[+,<,\mathrm{POW}_b] \leq_{\mathsf{N}} [S,<]^2$

We extend the results of the last section by adding predicates to test powers.

Lemma 7.1 *Let* $b \geq 2$. $[+,<,\mathrm{POW}_b] \leq_{\mathsf{N}} [S,<]^2$ *with parameters* (POW_b, f) *where* f *maps* b^i *to* i*.*

Proof:

Let $\phi(X)$ be a query over $[+,<]$. Our only concern is when $X \subseteq \mathrm{POW}_b$. We intend to replace X by a variable $\mathcal{X}$ such that as X ranges over POW_b, $\mathcal{X}$ ranges over N. The statement $b^i \in X$ will be coded by $bi+1 \in \mathcal{X}$.

We will replace the original first order variables with second order variables. These second order variables will be constrained to be finite sets that are also proper (to be defined later). We give some examples of how we do the coding before giving a formal definitions.

Example 7.2 Let $b = 5$. All strings mentioned are infinite with the unrepresented part being an infinite string of 0's going off to the left. These infinite strings will later be thought of as characteristic sequences for a set (reading from right to left, which is the reverse of the usual convention).

1. The infinite string

   $$A = \cdots \quad 10000 \quad 00001 \quad 00010 \quad 00100.$$

   will represent, in base 5, the number 4012. This is because the numbers in the 0th, 1st, 2nd and 3rd block are 2,1,0,4 respectively. We will also view A as representing a set. This set has $A(0) = 0$, $A(1) = 0$, $A(2) = 1$, $A(3) = 0$, and $A(4) = 0$. For this purpose we read the string from right to left, so we have that $A(x+1)$ is to the left of $A(x)$.

2. The infinite string

   $$\cdots \quad 00010 \qquad 00000 \quad 00000 \quad 00000 \quad 00000$$

   represents 10000 in base 5 which is 5^4 in base 10. More generally a string represents a power of 5 iff there is only one 1 in it, and that place is $n \equiv 1 \pmod 5$. Note that the string that represents 5^i to us looks like $5i+1$ the normal way of looking at strings.

3. The infinite string

$$\cdots \quad 10001 \quad 00000 \quad 00000$$

does not represent anything. Given a string we can test for this kind of property by insisting that if two bit places $x > y$ are both one then there is a number $z \equiv 0 \pmod 5$ such that $x \geq z > y$.

4. Any infinite string whose first five bits are 00100 must be $\equiv 2 \pmod 5$. More generally, for any b, the first b bits of x determine the i such that $x \equiv i \pmod b$.

Notation 7.3 Let $B_i = \{bi, \ldots, b[i+1]-1\}$.

Definition 7.4 A set Y is *proper* if Y is finite and $(\forall i)[|Y \cap B_i| \leq 1]$. For all i let m_i be the unique number in $Y \cap B_i$ if such a number exists, and bi otherwise (almost all of the m_i are bi). The *number coded by* Y is $\sum_{i=0}^{\infty}(m_i - bi)b^i$ and is denoted $code(Y)$.

We will be able to replace $+$ by S and quantifiers. We will then turn $\mathcal{X}$ back into a second order variable. At that point we will have a query in $[S,<]^2$.

Definition 7.5 Let MOD be the formula with prefix $(\exists C_0, \ldots, C_{b-1})$ and body

$$\bigwedge_{i=0}^{b-1} \left[(i \in C_i) \wedge (\forall x)[x \in C_i \Rightarrow (x + b \in C_i \wedge \bigwedge_{1 \leq j \leq b-1} x + j \notin C_i)] \right].$$

This formula asserts the existence of $C_0, \ldots, C_{b-1}$ such that, for $0 \leq i \leq b-1$, $C_i = \{n : n \equiv i \pmod b\}$. They will of course be helpful for coding the MOD_b functions, however they will also be helpful for ensuring that a finite set is proper.

Definition 7.6 Let $FIN(Y)$ be the formula that asserts that Y is finite (see Definition 6.4). Let $PROPER(Y)$ be the formula that asserts that Y is proper, namely

$$FIN(Y) \land MOD \land (\forall x > y)[(x \in Y \land y \in Y) \Rightarrow (\exists z)[z \in C_0 \land x \geq z > y]].$$

See the examples to see why this works.

Definition 7.7 Let $\mathrm{POW}_b(Y)$ be the formula that asserts that $\mathit{code}(Y) \in \mathrm{POW}_b$, namely

$$MOD \land (\exists n)[n \in Y \land (\forall m \neq n)[m \notin Y] \land n \in C_1].$$

$ONE(Y, Z)$ is the formula that states that $|Y| = 1$ and that one element is in Z, namely

$$(\exists x_1)[(x_1 \in Y) \land (x_1 \in Z) \land (\forall x_2 \neq x_1)[x_2 \notin Y]].$$

The following reduction transforms a formula ϕ of $[+, <, \mathrm{POW}_b]$ into a formula of $[S, <]^2$.
REDUCTION OF $[+, <, \mathrm{POW}_b]$ TO $[S, <]^2$, abbreviated RED

1. Input(ϕ).
2. Replace all occurrence of $s < t$ with $(\exists x)[x \neq 0 \land s + x = t]$. Hence the formula has no $<$ in it. Call the resulting formula ϕ.
3. If $\phi = \phi_1 \land \phi_2$ then output $\mathrm{RED}(\phi_1) \land \mathrm{RED}(\phi_2)$. If $\phi = \neg\phi_1$ then output $\neg\mathrm{RED}(\phi_1)$. If $\phi = (\exists y)[\phi_1(y)]$ then output $(\exists Y)[PROPER(Y) \land \mathrm{RED}(\phi_1(y))]$. (Make sure that when y becomes a second order variable that it is Y.) For the rest of this algorithm we assume ϕ is atomic.
4. ϕ is $n \in X$ for some natural number n. If $n \notin \mathrm{POW}_b$ then replace this with $1 \neq 1$. If $n = b^i$ then replace with $bi + 1 \in \mathcal{X}$.
5. ϕ is $y = n$ for some natural number n. Replace with $Y = \mathit{code}^{-1}(n)$.
6. ϕ is $y \in X$. Replace this with $\mathrm{POW}_b(Y) \land ONE(Y, \mathcal{X})$. We are saying that Y is a set that codes a power of b, say b^i (hence $bi + 1 \in Y$) and $b^i \in X$, coded by having $bi + 1 \in \mathcal{X}$.

7. ϕ is $\mathrm{POW}_b(y)$. Replace this with $\mathrm{POW}_b(Y)$.

8. ϕ is $y \equiv i \pmod{b}$. Replace this with $i \in Y$. (See the example above.)

9. ϕ is $y_1+y_2 = y_3$. We will replace y_1, y_2, y_3 with second order variables (if they ever get quantified over then they will be forced to be proper). We will obtain a formula that codes $\mathit{code}(Y_1)+\mathit{code}(Y_2) = \mathit{code}(Y_3)$. Let C be a new second order variable. C will represent bit carries. Let *place* and *carry* be the functions from $\{0,\ldots,b-1\}^3$ to $\{0,\ldots,b-1\}$ such that for $b_1, b_2, b_3 \in \{0,\ldots,b-1\}$ the sum $b_1+b_2+b_3$, when written in base b, is $\mathit{carry}(b_1,b_2,b_3)\cdot\mathit{place}(b_1,b_2,b_3)$. (Note that here and elsewhere $\cdot$ means concatenation, not multiplication.) Replace $y_1 + y_2 = y_3$ by the formula that begins with $(\exists C)(\forall i)$ and then has the conjunction of the following.

   (a) $(0 \notin C) \wedge (1 \notin C) \wedge \cdots \wedge (b-1 \notin C)$.

   (b) For every $(b_1, b_2, b_3) \in \{0,1\}^3$ the formula

$$[(i+b_1 \in Y_1) \wedge (i+b_2 \in Y_2) \wedge (i+b_3 \in C)] \Rightarrow \\ [(i+\mathit{place}(b_1,b_2,b_3) \in Y_3) \wedge (i+b+\mathit{carry}(b_1,b_2,b_3) \in C)]$$

10. Output the resulting formula.

END OF REDUCTION

a: Command not found. The proof that this is a reduction is similar to the last part of the proof of Lemma 6.1 and is hence omitted. ▮

Note 7.8 The proof of Lemma 7.1, when using $b = 2$, *does not* become the proof of Lemma 6.5. For $b = 2$ we could use 0 and 1 to code 0 and 1 in binary. The proof of Lemma 7.1, specialized to $b = 2$, would have coded 0 as 00 and 1 as 10.

8 $[S, <, P]^2 \le_{wb} [S, <]^2$ for several P

We show that, for several predicates P, $[S, <, P]^2 \le_{wb} [S, <]^2$.

Notation 8.1 Throughout this section P is computable, infinite, and co-infinite. We use P to denote the set P, the predicate that tests if numbers are in P, and the characteristic string of P.

Some of our techniques are based on [5].

Definition 8.2 Let $d \in \mathsf{N}$, Σ be a finite alphabet, $\sigma \in \Sigma$. The partial function $thin_{d,\sigma} : \Sigma^\omega \rightarrow \Sigma^\omega$ is defined as follows. If $X = \sigma^{p_1}\tau_1\sigma^{p_2}\tau_2\cdots$ ($p_1 = 0$ is allowed) where, for all i, $\tau_i \in \Sigma - \{\sigma\}$, then let

$$q_i = \begin{cases} p_i & \text{if } p_i \le d; \\ \mu q[d < q \le d + d! \wedge q \equiv p_i \pmod{d!}], & \text{otherwise} \end{cases}$$

then define

$$thin_{d,\sigma}(X) = \sigma^{q_1}\tau_1\sigma^{q_2}\tau_2\cdots.$$

If X is not of the correct form then $thin_{d,\sigma}(X)$ is undefined. This only occurs when $X \in \Sigma^*\sigma^\omega \cup \Sigma^*(\Sigma - \sigma)^\omega$.

The function $thin_{d,0}$ is a partial function from $\mathcal{P}(\mathsf{N})$ to $\mathcal{P}(\mathsf{N})$. The only elements not in the domain are finite and co-finite sets. The set $thin_{d,0}(P)$ will be very important.

Notation 8.3 If $\Sigma = \{0,1\}^m$ then we denote $A \in \Sigma^*$ by $(A_1, A_2, \ldots, A_m)$ where $A_i = \Pi_m(A)$. The term $thin_{d,00}(A, P)$ means that we have $\Sigma = \{00, 01, 10, 11\}$, A and P are subsets of N, and the input to $thin_{d,00}$ is the $x \in \Sigma^\omega$ such that $\Pi_1(x)$ is the characteristic string for A, and $\Pi_2(x)$ is the characteristic string for P. Note that 00 is a character in this context.

Proposition 8.4 (Theorem 3 of [5]) *Let Σ be a finite alphabet, σ be an element of Σ, $\mathcal{A}$ be a Büchi-automaton over Σ, and n be the number of states in N. Let $d \ge 2^{n^2 2^n}$. If X is in the domain of $thin_{d,\sigma}$ then ($\mathcal{A}(X)$ accepts iff $\mathcal{A}(thin_{d,\sigma}(X))$) accepts.*

From Proposition 8.4 and 5.5 we easily obtain the following.

Lemma 8.5 *Given $\psi(X_1, \ldots, X_m)$, a formula in $[S, <]^2$, one can effectively find a number d_0 such that, for all $d \ge d_0$,*

$$\psi(A_1, \ldots, A_m) \text{ iff } \psi(thin_{d,0^m}(A_1, \ldots, A_m)).$$

Definition 8.6 Let $d \in \mathsf{N}$ and $P = 0^{p_1}1^{r_1}0^{p_2}1^{r_2}\cdots$. Let $q_1, q_2, \ldots$ be as in Definition 8.2. Note that $thin_{d,0}(P) = 0^{q_1}1^{r_1}0^{q_2}1^{r_2}\cdots$. We define a computable bijection $f_{d,P}$ from P to $thin_{d,0}(P)$, with computable inverse, as follows. Let x be the input. If $x \notin P$ then diverge. (This can be tested since P is computable.) Otherwise:

1. Find a, b such that x is the ath element of the bth block of 1's in P. (Formally $x = a - 1 + \sum_{i=1}^{b-1}(p_i + r_i)$.)
2. Output the ath element of the bth block of 1's in $thin_{d,0}(P)$. (Formally $f_{d,P}(x) = a - 1 + \sum_{i=1}^{b-1}(q_i + r_i)$.)

We extend $f_{d,P}$ to subsets of P by $f_{d,P}(A) = \{f_{d,P}(x) : x \in A\}$.

Lemma 8.7 *Let $d \in \mathsf{N}$, $A \subseteq P$ and $f = f_{d,P}$. Then $thin_{d,00}(A, P) = (f(A), thin_{d,0}(P))$.*

Proof: Let $A \subseteq P$. Let

$$P = 0^{p_1}1^{r_1}0^{p_2}1^{r_2}\cdots.$$

Hence

$$A = 0^{p_1}\sigma_1 0^{p_2}\sigma_2\cdots$$

where $(\forall i)[|\sigma_i| = r_i]$. Therefore

$$f(A) = 0^{q_1}\sigma_1 0^{q_2}\sigma_2\cdots.$$

Since $A \subseteq P$, if $x \notin P$ then $x \notin A$. Hence if $x \notin P$ then 00 is in the xth place of (A, P). If $x \in P$ then clearly 00 is not in the xth place of (A, P). Hence 00 is in the xth place of (A, P) iff $x \notin P$, so

$$thin_{d,00}(A, P) = (0^{q_1}\sigma_1 0^{q_2}\sigma_2\cdots, 0^{q_1}1^{r_1}0^{q_2}1^{r_2}\cdots) = (f(A), thin_{d,0}(P)).$$

■

Lemma 8.8 *Let P be computable, infinite, and co-infinite. Assume that $\{thin_{d,0}(P)\}_{d=1}^{\infty}$ is well behaved (see Definition 4.6). Then $[S, <, P]^2 \le_{wb} [S, <]^2$ with parameters $(P, \{thin_{d,0}(P)\}_{d=1}^{\infty}, \{f_{d,P}\}_{d=1}^{\infty})$.*

Proof:
ALGORITHM

1. Input($\psi(X)$), a query in $[S, <, P]^2$.
2. Effectively create the formula that results in replacing the predicate P in ψ by free set variable Y and denote it by $\phi(X, Y)$.
3. Find d such that $\phi(X, Y)$ iff $\phi(thin_{d,00}(X, Y))$. (This is possible by Lemma 8.5.)
4. Find $\Delta_d(Y)$ such that $\Delta_d(A)$ iff $A = thin_{d,0}(P)$. (This can be done since $\{thin_{d,0}(P)\}_{d=1}^{\infty}$ is well behaved.)
5. Let $\psi^{\$}(X) = (\exists Y)[\phi(X, Y) \wedge \Delta_d(Y)]$. Note that this is equivalent to $\phi(X, thin_{d,0}(P))$.
6. Output($\psi^{\$}, d$).

END OF ALGORITHM

We need to show that if $A \subseteq P$ then $\psi(A)$ iff $\psi^{\$}(f_{d,P}(A))$. Let $A \subseteq P$.
$\psi(A)$ iff $\phi(A, P)$ by the definition of ϕ.
$\phi(A, P)$ iff $\phi(thin_{d,00}(A, P))$ by the definition of d.
$\phi(thin_{d,00}(A, P))$ iff $\phi(f_{d,P}(A), thin_{d,0}(P))$ by Lemma 8.5.
$\phi(f_d(A), thin_{d,0}(P))$ iff $\psi^{\$}(f(A))$ by definition of $\psi^{\$}$.
▮

We can now use Lemma 8.8 and results from the literature to show that several languages reduce to $[S, <]^2$.

Lemma 8.9 *Let* $P \in \{\mathrm{FAC}, \mathrm{POLY}_b, \mathrm{POW}_b : b \in \mathsf{N}\}$ *and* $L = [S, <, P]^2$. *Then* $L \leq_{wb} [S, <]^2$.

Proof: Let $P \in \{\mathrm{FAC}, \mathrm{POLY}_b, \mathrm{POW}_b : b \in \mathsf{N}\}$. By [5, Theorem 4] $\{thin_{d,0}(P)\}_{d=1}^{\infty}$ is well behaved. Hence, by Lemma 8.8, $L \leq_{wb} [S, <]^2$. ▮

9 Corollaries

Corollary 9.1 *Let L be any of the following languages.*

- $[S,<]^2$.
- $[+,<]$.
- *For any b,* $[+,<,\mathrm{POW}_b]$.

Let $a,d,n \in \mathsf{N}$ such that $d \geq 1$. Then the following are true.

1. $COMP \notin QBC^a[L]$.
2. $COMP \notin [1,d]QBC^a[L]$.
3. $PEX_{n+1} - QEX_n[L] \neq \emptyset$.
4. $QEX_n[L] \subset QEX_{n+1}[L]$.
5. *L is QCD.*

Proof: Clearly $[S,<]^2 \leq_{\mathsf{N}} [S,<]^2$. By Lemmas 6.1 and 7.1 $L \leq_{\mathsf{N}} [S,<]^2$ for $L = [+,<]$ or $[+,<,\mathrm{POW}_b]$. Hence for these languages $COMP \notin QBC^a[L]$ by Theorem 5.17, $COMP \notin [1,d]QBC^a[L]$ by Corollary 5.20, $PEX_{n+1} - QEX_n^a[L] \neq \emptyset$ and $QEX_n[L] \subset QEX_{n+1}[L]$ by Theorem 5.23, and L is QCD by Theorem 5.28. ▮

Corollary 9.2 *Let L be any of the following languages.*

- *For any b,* $[S,<,\mathrm{POW}_b]^2$.
- *For any b,* $[S,<,\mathrm{POLY}_b]^2$.
- $[S,<,\mathrm{FAC}]^2$.

Let $a,d,n \in \mathsf{N}$ such that $d \geq 1$. Then the following are true.

1. $PEX_{n+1} - QEX_n[L] \neq \emptyset$.
2. $QEX_n[L] \subset QEX_{n+1}[L]$.

3. *L is QCD.*

Proof: By Lemma 8.9 if L is $[S, <, \mathrm{POW}_b]^2$, $[S, <, \mathrm{POLY}_b]^2$, or $[S, <, \mathrm{FAC}]^2$, then $L \leq_{wb} [S, <]^2$. Hence for these languages $PEX_{n+1} - QEX_n^a[L] \neq \emptyset$ and $QEX_n[L] \subset QEX_{n+1}[L]$ by Theorem 5.26, and L is QCD by Theorem 5.27. ▮

10 Further Results

In this paper we showed how to take results about queries to $[S, <]^2$ and extend them to different languages L. However, obtaining results about $[S, <]^2$ may still require some effort. In a subsequent paper [9] a general theorem was proven that (roughly) allows you to obtain results about query learning with $[S, <]^2$ from results about passive learning. Combining there results with ours we obtain the following.

If $L \leq [S, <]^2$ then the following hold.

1. $QEX^1[L] \subset QEX^2[L] \subset \cdots$.
2. $EX^{a+1} - QEX^a[L] \neq \emptyset$.
3. $QBC^1[L] \subset QBC^2[L] \subset \cdots$.
4. $BC^{a+1} - QBC^a[L] \neq \emptyset$.
5. $QEX[L] \subset [1,2]QEX[L] \subset [1,3]QEX[L] \cdots$.
6. $QBC[L] \subset [1,2]QBC[L] \subset [1,3]QBC[L] \cdots$.
7. $EX_0^* - [1,d]QEX^a[L] \neq \emptyset$. (This had appeared in an earlier version of this paper with a very complicated proof.)

11 Open Questions

We have that $[S, <, \mathrm{POW}_2]^2$ and $[S, <, \mathrm{POW}_3]^2$ are QCD but what about $[S, <, \mathrm{POW}_2, \mathrm{POW}_3]^2$? This question is somewhat premature since it is not even known if (first order) $[S, <, \mathrm{POW}_2, \mathrm{POW}_3]$ is decidable.

12 Acknowledgement

We would like to thank Kalvis Apsītis, Mark Bickford, Andrew Lee, and Mark Pleszkoch, for proofreading and discussion.

References

[1] J. R. Büchi. Weak second order arithmetic and finite automata. *Zeitsch. f. math. Logik und Grundlagen d. Math.*, 6:66–92, 1960.

[2] J. R. Büchi. On a decision method in restricted second-order arithmetic. In *Proc. of the International Congress on logic, Math, and Philosophy of Science (1960)*, pages 1–12. Stanford University Press, 1962.

[3] J. Case and C. H. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Comput. Sci.*, 25:193–220, 1983.

[4] Y. Choueka. Theories of automata on ω-tapes: A simplified approach. *Journal of Computer and Systems Sciences*, 8:117–142, 1974.

[5] C. C. Elgot and M. O. Rabin. Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. *Journal of Symbolic Logic*, 31(2):169–181, June 1966.

[6] Y. Ershov, I. Lavrov, A. Taimanov, and M. Taitslin. Elementary theories. *Russian Math Surveys*, 20:35–105, 1965.

[7] R. Freivalds, C. H. Smith, and M. Velauthapillai. Trade-off among parameters affecting inductive inference. *Information and Computation*, 82(3):323–349, Sept. 1989.

[8] W. Gasarch, E. Kinber, M. Pleszkoch, C. H. Smith, and T. Zeugmann. Learning via queries, teams, and anomalies. *Fundamenta Informaticae*, 23:67–89, 1995. Earlier version in COLT90, pages 327–337.

[9] W. Gasarch and A. C. Lee. Inferring answers to queries. In *Proceedings of* 10^{th} *Annual ACM Conference on Computational Learning Theory*, pages 275–284, 1997. Long version on Gasarch's home page.

[10] W. Gasarch, M. Pleszkoch, and R. Solovay. Learning via queries to $[+, <]$. *Journal of Symbolic Logic*, 57(1):53–81, Mar. 1992.

[11] W. Gasarch and C. H. Smith. Learning via queries. *Journal of the ACM*, 39(3):649–675, July 1992. Earlier version is in 29th FOCS conference, 1988, pp. 130–137.

[12] W. Gasarch and M. Velauthapillai. Asking questions versus verifiability. *Fundamenta Informaticae*, pages 1–9, 1997.

[13] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:522–530, 1966.

[14] D. Osherson, M. Stob, and S. Weinstein. Aggregating inductive expertise. *Information and Computation*, 70:69–95, 1986.

[15] M. O. Rabin. Decidable theories. In J. Barwise, editor, *Handbook of Mathematical Logic*. North Holland, 1977.

[16] S. Safra. On the complexity of ω-automata. In *Proc. of the 29th IEEE Sym. on Found. of Comp. Sci.*, pages 319–329, 1988.

[17] A. L. Semenov. On certain extensions of the arithmetic of addition of natural numbers. *Mathematics of the USSR–Izv*, 15:401–418, 1980. ISSN 0025-5726.

[18] C. H. Smith. The power of pluralism for automatic program synthesis. *Journal of the ACM*, 29(4):1144–1165, October 1982. Earlier verssion in FOCS 1981, pages 283–295.

[19] F. Stephan. Learning via queries and oracles. *Annals of Pure and Applied Logic*, 94:273–296, 1998. Earlier version in COLT95, pages 162–169.

[20] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 757–804. MIT Press and Elsevier, The Netherlands, 1990.